CENA 10.000 zł

ISSN 0867-3918 INDEKS 377112

64 PLUS 4 & AMIGA

2/'92

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



Drodzy Czytelnicy!

W związku z unifikacją sieci informatycznej, jaką ostatnio wprowadziły banki w Polsce, otrzymaliśmy **nowy numer konta!** Zwracamy na to szczególną uwagę! Na trzeciej stronie okładki, w reklamie naszego rocznika, umieszczony jest jeszcze stary numer konta - informacja o jego zmianie dotarła do nas, kiedy okładka była już wydrukowana i nie mieliśmy możliwości wprowadzenia zmiany. Mamy zapewnienie z Banku S.A., iż wszystkie wpłaty, na których umieszczony będzie stary numer konta, dotrą do nas bez problemu.

Informujemy również, że nasze pismo można w dalszym ciągu **zaprenumerować** - co daje pewność systematycznego otrzymywania (drogą pocztową). Nasz miesięcznik kosztuje w prenumeracie 10.000 zł. Prenumeratę można zawrzeć na okres nie krótszy niż dwa miesiące, w dowolnym okresie, maksymalnie do końca roku kalendarzowego. Wykupujący prenumeratę nie ponoszą kosztów przesyłki pocztowej.

REDAKCJA

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych do prowadzenia kolportażu „64 plus 4 & Amiga" (kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.



miesięcznik nr 2(16)
luty 1992
cena 1 egz.: 10.000 zł.
64 PLUS 4

WYDAWCA: ABUK Spółka z o.o.
REDAGUJĄ: Waldemar Szczygieł (redaktor naczelny) z zespołem.
ADRES REDAKCJI: Redakcja „64 plus 4", 85-166 Bydgoszcz 43, skrytka pocztowa 64.
OKŁADKA: Piotr Bartz.
SKŁAD: ABUK
DRUK: W.Z.G. Wąbrzeźno, okładka: Z.P. POLRASTER, Bydgoszcz.

W numerze :



COMMODORE w Polsce!

30 stycznia, w Warszawie, w hotelu Mariott odbyło się otwarcie oficjalnego przedstawicielstwa firmy Commodore na nasz kraj. Członkowie redakcji „64 plus 4 & Amiga" byli obecni na tej uroczystości.

Firma Commodore zamierza wesprzeć rozwój gospodarczy Polski uważając, że warunki rozpoczęcia efektywnej współpracy są sprzyjające. Potrzeba modernizacji i skłonności polskich przedsiębiorców do inwestowania w nowoczesną technologię komputerową jest głęboka, jak nigdy dotąd. Tym bardziej, że po likwidacji wschodnioniemieckiego producenta komputerów Robotron, którego produkty zajmowały przodujące miejsce na rynku PC w Polsce i Europie Wschodniej, rynek ten stoi przed całkowitą odbudową. Commodore wraz ze swoją kompetencją w dziedzinie komputerowej i doświadczeniem w zakresie know-how stawia czoła temu wyzwaniu. Duże przedsiębiorstwa, zakłady średnie i drobne, jak również szkoły, banki, ubezpieczalnie oraz gminy sięgają w większym stopniu po komputery Commodore.

Aby optymalnie sprostać tym potrze ›om, Commodore już teraz wchodzi na polski rynek z całym swoim asortymentem. W dziedzinie szkolnictwa i kształcenia zawodowego firma oferuje modele C 64 i Amiga, a w dziedzinie przemysłowej profesjonalne modele PC i rozwiązania Network.

Właściwa historia firmy Commodore zaczyna się w 1960 r. wraz z założeniem „Commodore Business Maschines Inc." w Westchester/Pensylwania. W latach sześćdziesiątych w amerykańskich przedsiębiorstwach do nowoczesnych metod rozliczeń administracyjnych zaczęto stosować kalkulatory. Do tej pory zbierano duże ilości danych i przetwarzano je ręcznie w tych wypadkach, kiedy było to opłacalne. Użytkownicy w pewnych gałęziach handlu, np. w bankach, na giełdach, jak również w przedsiębiorstwach średnich, zaczęli domagać się coraz bardziej systemów, które byłyby w stanie przetwarzać rosnący napływ informacji, w momecnie ich powstawania , oraz dostarczać ich do natychmiastowego użytku.

W 1967 r. przyszedł na to czas: Commodore wypuszcza na światowy rynek pierwszy elektroniczny kalkulato.

Dla dalszego rozwoju tej technologii Commodore założyło w 1969 r., w legendarnej dziś Silicon Valley, pierwsze laboratorium rozwojowe.

W następnych latach dał się zauważyć przyspieszoıy rozwój technik przetwarzania infoı nacji. Commodore w porę wyczuło ważny trend: oddzielono wydział mikrokomputerowy od wydziału systemów profesjonalnych. Wskutek tego wypuszczono na rynek, komercyjną serię 8000, skonstruowaną ze starego PET 2001. W dziedzinie mikrokomputerów firma wypuściła nowy VC 20. Jego możliwości w wyczarowywaniu kolorowych obrazów i grafik na ekranie monitora przyczyniły się do tego, że w ciągu kilku miesięcy zdobył on rynek światowy. Do roku 1982 sprzedano ponad dwa miliony urządzeń.

Teraz wydarzenia następowały szybko, jedno za drugim: w roku 1982 firma prezentuje kolejną światową nowość: Commodore 64. Wraz z nim technologia komputerowa stała się dostępna dla każdej kieszeni. Dla wielu okazał się dobrym komputerem uniwersyteckim, znalazł również szerokie zastosowanie w wielu przedsiębiorstwach; nawet dzisiaj, pod względem rozdzielczości koloru i grafiki, jest nadal dobrym sprzętem dla początkujących. Tylko w 1990/91r. sprzedano ponad 400.000 sztuk C 64.

Bez wątpienia - aktualnie - ponad dziesięć milionów użytkowników, w ponad stu krajach, uczyniło C 64 największym sukcesem komputerowym na świecie.

W 1985 r. po skutecznym wejściu na rynek MS i DOS, Commodore wywołało sensację dwiema nowościami: serią produkcyjną PC i Amigą. Wejście w odpowiednim momencie zarówno na rynek profesjonalny, jak i na rynek masowego użytkownika - okazuje się dzisiaj słusznym posunięciem.

Commodore zajmuje od kilku lat przodujące miejsce na niemieckim rynku mikrokomputerowym. Według najnowszych informacji angielskiego instytutu badań rynku Dataquest, Commodore zajmuje w tej chwili drugie miejsce na europejskim rynku PC.

W roku 1990 przekroczono dwa miliony sprzedanych egzemplarzy Amigi. W zeszłym roku sprzedano trzy milionową sztukę. Jeszcze w tym roku firma spodziewa się przekroczenia granicy czterech milionów sprzedanych urządzeń.

Następnym przykładem sukcesu strategii marketingu Commodore jest niedawne wprowadzenie na rynek światowej nowości: CDTV. „Commodore Dynamic Total Vision" jest pierwszym urządzeniem umożliwiającym łączenie CD-ROM z technologią Amigi i dostarczenia dotychczas nieznanej jakości Multimediów poprzez wzajemne uzupełnianie się obu systemów.

Po PET 2001, VC 20, C 64 i Amidze także i „Profi-Line" wskazuje nową drogę, tym razem dla rynku Multimediów lat dziewięćdziesiątych. Od kilku miesięcy liczące się czasopisma fachowe piszą o nowym systemie CDTV, który dla wielu ludzi już teraz stanowi zupełnie nowy środek masowego przekazu w użytkowej dziedzinie Multimediów.

W poprzednich częściach artykułu opisywaliśmy zakres i działanie zmiennych. Dziś komputer pokaże nam jak administruje programem.

C-16

Uczymy się programować (cz. 5)

Adres początku pamięci programu zapamiętany jest w komórkach 43 i 44 (na stronie zerowej) i można go odczytać za pomocą rozkazu:

print peek(43)+256*peek(44).

W przypadku C-16 daje to zawsze wynik 4097, niezależnie od tego czy załączona jest grafika wysokiej rozdzielczości, czy też nie. W przypadku Plus 4 pamięć programu w trybie grafiki, rozpoczyna się od 16385 i można ją przesunąć z powrotem do 4097 używając rozkazu **graphic clr**.

Wskazówka 22:
Rozkaz **graphic 0** przełącza tylko ekran z grafiki w tryb tekstowy, nie zmienia natomiast podziału pamięci. Rozkaz **graphic clr** czyni wolnym 10 kbajtów pamięci, zarezerwowanej dla grafiki i przesuwa przy Plus 4 początek Basic'a z powrotem do 4097.

Aby umożliwić obserwację pracy komputera - tradycyjnie już - pamięć programu zostanie „przeniesiona" na ekran (adres początkowy 3072, starszy bajt 12).

Wykonajmy następujące kroki:
1. Przełączamy komputer w tryb pisania małych liter. Ułatwia to odszyfrowanie kodu ekranu.
2. Przenieśmy fragment pamięci na ekran za pomocą: - **poke 44, 12: new.** New powoduje odpowiednie przesunięcie pamięci zmiennych. Sygnał błędu **syntax error** (błąd składni) możemy zignorować.
3. Kasujemy ekran.
4. Umieszczamy w lewym górnym rogu ekranu trzy klamry.
5. Nie używając Return'u przesuwamy kursor dwie linie niżej.

Trzy klamry zastępują w kodzie ekranu trzy zera, z których składa się pusty program w Basic'u. Po tych zabiegach komputer jest gotowy do wprowadzenia pierwszej lini programu:

1 rem proba

Po użyciu Return pojawia się w górnej lini ciąg oznaczeń, z którego można od razu odszyfrować wyraz PROBA. Jest on napisana dużymi literami ponieważ instrukcja została zapamiętana w kodzie ASCII. Małe litery mają w kodzie tym numery od 65 do 90. Kod ekranu interpretuje te liczby jako duże litery.

Co dzieje się z **rem**. Komputer rozpoznał je jako słowo rozkazu i zamiemienił w symbol - liczbę między 128 i 255.

Symbolem (token - patrz oryginalna instrukcja obsługi) dla **rem** jest liczba 143 (=128+15), a więc małe o w inwersji, które zlokalizowane jest na lewo od PROBA. Na rysunku 8 jest też przedstawiona budowa linii Basic'a. Na początku jest zero, po którym następuje Link-Pointer (wskaźnik połączenia). Podaje on w dobrze znanej formie LOW/HIGH, gdzie zaczyna się następna linia. Umieszczony tutaj młodszy bajt m=13, a starszy l=12. Wskaźnik ten wskazuje więc na adres 13+256*12=3085, tzn. na drugie z trzech zer (klamr), które ten program kończą. Od tego miejsca rozpoczyna się nowy adres połączenia, po wprowadzeniu dalszej linii.

0	1	2	3	4	5. Bajt...
zero początkowe	low	high	low	high	zawartość linii
	linkpointer		numer linii		

Rys. 8. Budowa linii w Basic'u

W obu następnych bajtach (a=1 i klamra) jest zaszyfrowany numer linii 1+256*0=1. Za pomocą dwóch bajtów można przedstawić 255+256*255=65535 liczb. Najwyższy dopuszczalny numer linii jest jednak trochę niższy (63999), co można łatwo sprawdzić.

Dopiszmy do naszego programu nową linię:

258 stop.

Program kończą znowu trzy klamry i obie linie są zastąpione przez zero. Nowy wskaźnik połączenia wstawia r i 1 razem i pokazuje znowu na drugie końcowe zero. Bajty numeru linii noszą nazwę b i a, tzn, że numer wynosi 2+256*1=258. Wyraz rozkazu **STOP** został zamieniony (małe p w inwersji - 128=16=144).

Co się dzieje, gdy w linii programu jest więcej rozkazów? Wpiszmy np.:

3 list: get: new.

C-16

Najpierw zwraca uwagę, że nowa linia 3, którą łatwo rozpoznać po dwukropku i trzech odwróconych symbolach, została uszeregowana przed linią 258. Warte zauważenia jest także to, że dla obu dodatkowych rozkazów nie użyto nowych numerów linii, wskaźników połączenia czy rozdzielających zer. Przez to połączenie rozkazów w linii 3 zaoszczędzono 10 bajtów. Także przebieg programu jest szybszy, ponieważ komputer przy odszukiwaniu określonej linii skacze od linkpointer'a (od łącznika do łącznika).

Wskazówka 23:
Jeśli zależy nam na zaoszczędzeniu czasu i pamięci, powinniśmy w jednej linii umieścić jak najwięcej rozkazów. Naturalnie pogarsza to przejrzystość listingu.

Administrowanie liniami zostanie wyjaśnione dzięki następującemu programowi. Najpierw kasujemy ekran, a przez to także stary program i w górnym rogu pojawiają się znowu trzy klamry. Od trzeciej linii podaje się:

```
3 rem3
2 rem2
1 rem1
```

Widać wyraźnie, że komputer ma trudności z uporządkowaniem linii, ponieważ każdorazowo muszą zostać zmienione adresy link. Aby zmniejszyć manipulowanie numerami linii, należy kursor umieścić w górnej linii, za pomocą klawisza HOME (bez SHIFT!) i w pierwszym rozkazie zmienić numer z **a** (=1) na **c** (=3). Bez Return'u kursor przesuwamy do trzeciej linii i program jest listujemy. Są teraz dwie różne linie o numerze 3, a program można jeszcze listować! Jak to funkcjonuje? Komputer „zawiesza się" przy listowaniu z jednego adresu - link do następnego i każdorazowo drukuje oba kolejne bajty jako numer linii. Można to także zobaczyć, kiedy spis zawartości dyskietki załaduje się za pomocą **DLOAD„$"** (np. aby go wydrukować) i wylistuje. Długości bloku będą przy tym widoczne jako liczby linii. Kiedy natomiast chcemy wylistować pojedynczą linię np. za pomocą **list2**, to komputer jej nie odnajdzie, ponieważ przerywa poszukiwanie linii 2 po odnalezieniu linii 3.

O ogromie pracy jaką wykonuje komputer można się także przekonać po zmianie linkpointer'a. Najlepiej sprawdzić to wpisując raz jeszcze trzy linie **REM**. Wskaźniki połączenia nazywają się **hl**, **ol i vl**. Po zastąpieniu **hl** przez **ol**, przy listowaniu znika druga linia. Zupełnie wyprowadza się komputer z równowagi po zastąpieniu ostatniego wskaźnika **vl** przez **ol**.

Po zastąpieniu pierwszego adresu Link **hl** przez dwie klamry, przy listowaniu stwierdzamy kompletną pustkę i program jest dla komputera stracony. Komputer nie szuka dalej trzech zer. Jest to sytuacja identyczna tak jak w przypadku NEW. Program nie jest skasowany lecz „zawieszony". Dlatego można go „powrócić do życia". Chociaż Basic 3.5 nie zawiera rozkazu **OLD** lub **RENEW** można za pomocą następującego triku, program po NEW, uratować.

Wskazówka 24
Jeśli omyłkowo poda się NEW, to można program uratować w następujący sposób (nie mogą być jednak zdefiniowane żadne zmienne):

```
poke 4997,1: delete 1.
```

Jeśli w przypadku Plus/4 nie wiadomo dokładnie gdzie leży początek Basic'a, to oznajmi nam to komputer po podaniu:

```
poke(peek(44)*256+1),1: delete 1.
```

Pierwszy rozkaz mówi komputerowi: zajrzyj do komórki pamięci 44 i liczbę, którą zawiera bajt Hight początku Basic'a, pomnóż przez 256 i dodaj 1.

Jak funkcjonuje teraz „czarodziejska formuła" ze wskazówki 23? Rozkaz **POKE** przywołuje z procedury systemu operacyjnego pewien ciąg, między innymi procedurę Link, która po skasowaniu linii oblicza nowy adres Link. Adres ten, zagubiony po podaniu **NEW** zostanie odnaleziony: komputer zagląda na pozycję pamięci 43/44, gdzie znajduje się początek Basic'a, odczytuje następne 4 bajty (w numerze linii, może znajdować się także zero), i szuka pierwszego zera rozdzielającego, po czym zapisuje jego adres podwyższony o jeden za zerem początkowym. Można to dobrze zaobserwować na ekranie. Wychodząc z programu z trzema liniami **REM**, wpisujemy **NEW**. Widać jak pierwszy adres Link jest zastępowany przez dwie klamry. Wykonując teraz **LIST** można sprawdzić czy program przepadł. Podaje się teraz najpierw:

```
poke 3073,1
```

(początek Basic'a przeniesiony jest teraz do pamieci ekranu, i rozpoczyna się adresem 3072). Na drugim miejscu znajduje się **a** (=1). Można zresztą za pomocą **POKE** podać każdą inną liczbę (z wyjątkiem zera).

Tak samo w drugim rozkazie można użyć każdą dowolną liczbę, np. **delete 1000**. Robimy **LIST** i okazuje się, że program jest całkowicie do dyspozycji. Przy podawaniu tych rozkazów, trzeba uważać, aby nie wpisać ich w dolną linię ekranu, ponieważ po Return'ie usuwana jest z ekranu górna linia programu.

Co zrobić, gdy po niezręcznym wczytaniu komputer „wysiada" i nie można zastosować „czarodziejskiej formuły"?. I na taką sytuację jest pewien trik...

Wskazówka 25
Kiedy komputer „zawiesił się", a więc nie ma kursora, należy - trzymając naciśnięty klawisz RUN/STOP - wcisnąć mały biały klawisz **Reset**, aż zamelduje się TEDMON. Po użyciu komendy **X** (+Return), wraca się wtedy do normalnego trybu bez utraty programu. Jeśli nacisnęlismy przez pomyłkę tylko klawisz Reset, to można taki program uratować, za pomocą wskazówki 23.

Po **NEW** i **Reset** istnieją więc także możliwości ratunku, ale co się dzieje, gdy skasuje się linię przez podanie jej numeru i naciśnięcie klawisza Return? Także i taką sytuację można wypróbować na naszej modelowej pamięci programu. Taka linia znika całkowicie z pamięci. Można mieć tylko nadzieję, że znajduje się ona przypadkowo na ekranie i za pomocą Return można ją na nowo odtworzyć. Ekran służy wtedy jako „międzypamięć". Na tym opiera się bardzo prosta metoda MERGE, która nadaje się wprawdzie tylko do „przyczepienia" bardzo krótkich programów (np. podprogramu) do programu głównego.

Wskazówka 26:

Procedura Merge dla dołączenia krótkich programów.

1. W krótkim programie zmienić wszystkie numery linii w ten sposób, aby na pewno leżały powyżej linii programu głównego.
2. Wylistować krótki program. Może obejmować nie więcej niż 20 linii ekranu.
3. Tuż pod ostatnią linią podać:
   dload„(nazwa programu głównego)" (return).
4. Nie listować głównego programu. Umieścić kursor za pomocą klawisza HOME (bez SHIFT!) w górnej linii i tak długo używać klawisz Return, aż wszystkie linie krótkiego programu zostaną podane na nowo.

Wróćmy jeszcze do słów rozkazu i ich symboli.

Po przeniesieniu pamięci programu na ekran (kroki od 1 do 5), dopiszmy:

1 end for next.

W górnej linii są widoczne trzy odwrócone oznaczenia, którym na podstawie tabeli kodu ekranu można przyporządkować liczby 128, 129 i 130. Chodzi więc o symbole trzech pierwszych słów rozkazu. Wprowadźmy teraz:

2 rem stop on.

Dziwne, STOP i ON jest już rozszyfrowane, podczas gdy zamiast REM, jak można było oczekiwać, pojawia się o w inwersji. Po REM komputer nie domyśla się dalszych rozkazów. Tak samo się dzieje, gdy zamknie się słowo rozkazu w cudzysłów. Także wtedy są one dla niego tekstem prostym. Skróty rozkazu mają te same symbole (token'y) co odpowiadające im rozkazy, dlatego przy listowaniu pojawiają się wpisane całe słowa.

Wskazówka 26

Przez zmianę miejsca pamięci 1177, można się przełączyć z RAM'u w ROM (Bankswitching - przełączenie banku), tak aby za pomocą **PEEK** można odczytać system pracy.

poke 1177,62:ROM

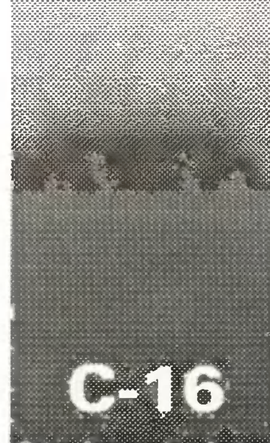
poke 1177,63:RAM (stan normalny)

Listing 1 pokazuje jak można za pomocą PEEK wszystkie słowa rozkazów Basic'a 3.5 odczytywać i przywoływać na ekran. Widzimy, że słowa rozkazów są w pamięci bez odstępów. Ostatnią literę poznaje się zawsze po tym, że do jej normalnej wartości ASCII, została dodana liczba 128, przez co na ekranie pojawia się ona w inwersji.

```
10  REM LISTA ROZKAZOW ZAWARTA W ROM
20  :
30  POKE 1177,62: REM WLACZENIE ROM
40  C=33166: REM ADRES POCZATKOWY LISTY
    ROZKAZOW
50  PRINT CHR$(147) CHR$(14)
60  FOR I=0 TO 499
70  A=PEEK (C+1): PRINT CHR$(A):
80  NEXT
90  POKE 1177,63: REM WLACZENIE RAM
```

Listing. 1.

Opr. na podst. RUN, nr. 6/81.



OGŁOSZENIA

C-64

Na Commodore 64 istnieje bardzo wiele programów służących do tworzenia obrazów w różnych trybach graficznych, jednak nie istnieje jeden standart ich zapisu. Dlatego też często występują problemy, gdy w swoim własnym programie zechcemy wykorzystać wcześniej narysowany obrazek. W tym artykule zajmiemy się wyświetlaniem obrazków w trybie wielokolorowym (MULTICOLOR) narysowanych przy pomocy najpopularniejszych programów graficznych.

Obrazki, obrazki...

Nasz obrazek będziemy starali się umieszczać w pamięci od adresu 8192 ($2000). Zaletą tego rozwiązania jest to, że nie trzeba zmieniać banków, z których korzysta VIC, natomiast wadą , że na nasz progam w BASIC'u zostaje tylko 6kB pamięci, a w kilku przypadkach nawet nieco mniej. Bardziej zawansowani użytkownicy mogą pokusić się o umieszczenie obrazka od adresu $C000, co umożliwia korzystanie z całych 38kB dostępnych dla BASIC'a.

Ale wróćmy do naszego prostszego rozwiązania. Bez względu na sposób zapisu obrazka na dysku tuż przed jego wyświetleniem należy wykonać następujące instrukcje:

```
10  POKE  53265,59
20  POKE  53270,216
30  POKE  53272,24
```

Linia 10 to włączenie trybu grafiki wysokiej rozdzielczości, linia 20 włączenie trybu wielokolorowego, natomiast linia 30 ustawia adres mapy bitowej obrazka na adres 8192, a danych o kolorze atramentu na adres 1024.

Następnie należy przy pomocy prostej pętli przepisać do odpowiednich obszarów pamięci dane o kolorach obrazka.

```
40  FOR I=0 TO 999
50  POKE   1024+I,PEEK(stala1+I)
60  POKE   55296+I,PEEK(stala2+I)
70  NEXT I 80 POKE 53280,PEEK(stala3)
90  POKE   53281,PEEK(stala3)
100 GOTO100
```

Linia 50 zajmuje się przepisywaniem danych o pierwszym i drugim kolorze atramentu do aktualnej pamięci ekranu, a linia 60 danych o trzecim kolorze atramentu do pamięci koloru. W liniach 80 i 90 zostają wpisane odpowiednie wartości do rejstrów koloru ramki oraz tła. W linii 100 umieszczona jest pętla nieskończona, dzięki której kolory obrazka nie będą zapsute przez napis READY pojawiający się po wykonaniu programu.

Pozostaje jeszcze wczytanie obrazka w odpowiednie miejsce pamięci. Niestety nie wszystkie programy graficzne nagrywają obrazki od adresu 8192. Dlatego też najlepiej do wczytywania ich do pamięci komputera posłużyć się dowlnym monitorem języka maszynowego (instrukcja L"nazwa",08,adres). Jeżeli nie posiadamy żadnego monitora to możemy skorzystać z następującej sekwencji rozkazów: POKE 43,adres-INT(adres/256) *256:POKE44,adres/256 oraz LOAD "nazwa_obrazka",8 i na końcu POKE 43,1: POKE 44,8: NEW.

Po wykonaniu tych instrukcji dane obrazka zostaną załadowane do pamięci poczynając od komórki "adres". Ostatnia sekwencja instrukcji przywraca standardowy adres początku BASIC'a, powoduje jednak utratę znajdującego się w pamięci programu. Tak więc zawsze należy najpierw wgrać obrazek, a dopiero potem program, w którym go chcemy wykorzystać. Przy czym należy zwrócić uwagę na to, aby nasz program nie przekraczał komórki **adres**, gdyż spowoduje to uszkodzenie samego obrazka. Teraz, aby wyświetlić dowolny obrazek wystarczy zamiast wyrażeń stala1, stała2 i stala3 w naszym progrmie podstawić odpowiednie wartości. Oto dane dla najpopularniejszych programów graficznych:

- KOALA MICROILUSTRATOR. Oryginalny adres obrazka to 25576 ($6000), jednak my wczytamy go od adresu 8192 ($2000), czyli we wcześniej podanej sekwencji rozkazów wpisujemy zamiast "adres" liczbę 8192. Po wczytaniu obrazka, od adresu 8192 do 16192 ($3F40) znajduje się jego mapa bitowa, od 16192 dane dla pierwszego i drugiego koloru atramentu, a od 17192 ($4328) dla trzeciego. Jeszcze tylko kolor tła, który jest umieszczony pod adresem 18192 ($4710). Tak więc do programu należy wstawić następujące wartości: **stala1=16192, stala2=17192, stala3=18192**. W chwilę po uruchomieniu programu na ekranie ukaże się nam obrazek w całej okazałości.

- BLAZZING PADDLES. Przy zwykłym ładowaniu obrazek wczytuje się od adresu $A000 (40960), my wczytujemy od 8192 (adres=8192). Mapa bitowa umieszczona od adresu 8192 do 16192, dane pierwszego i drugiego koloru atramentu od 16384 ($4000), a trzeciego od 17408 ($4400). Kolor tła znajduje się pod adresem 16256 ($3F80). Wartości stałych są więc następujące: **stala1=16256, stala2=17408, stala3=16256**. Pozostaje jeszcze tylko uruchomienie programu.

- ARTIST 64. Podobnie jak poprzednie obrazki, tak i ten ładujemy od adresu 8192. Rozmieszczenie danych dla wszystkich kolorów atramentu jest takie samo jak dla programu BLAZZING PADDLES. Różni się tylko adresem, w którym przechowywany jest kolor tła. Tak więc stałe mają następujące wartości: **stala1=16256, stala2=17408, stala3=18408** ($47E8).

• ADVANCED ART STUDIO. Adres ładowania obrazków w standarcie tego programu jest równy 8192, tak więc nie ma potrzeby korzystania z naszej sekwencji rozkazów służących do ładowania obrazków. Dane o pierwszym i drugim kolorze atramentu umieszczone są od adresu 16192, natomiast o trzecim od 17208 ($4338). Kolor tła zapamiętany jest w komórce 17193 ($4329). A oto wartości stałych: **stala1=16192, stala2=17208, stala3=17193.**

• IMAGE SYSTEM. Tym razem adres ładowania wynosi 7168 ($1C00). Pierwszy i drugi kolor atramentu umieszczony jest od adresu 16384, a trzeci od 7168. Kolor tła jest umieszczony w komórce 8168 ($1FE8). Wartości stałych muszą być następujące: **stala1=16384, stala2=7168, stala3=8168.**

• VIDCOM 64. Adres ładowania w przypadku tego programu jest najmniejszy i wynosi 6144 ($1800). Z tego powodu dla programu w Basic'u pozostawione jest tylko 4096 bajtów. Dane o pierwszym i drugim kolorze atramentu są umieszczone od adresu 7168, a o trzecim od 6144. Kolor tła jest zapamiętany w komórce 8168. Tak więc stałe mają wartości: **stala1=8168, stala2=6144, stala3=7168.**



Są to dane o popularniejszych programach graficznych dostępnych w Polsce. Mam nadzieję, że wszyscy znający choć trochę język maszynowy będą w stanie bez żadnego problemu stworzyć własne procedury wyświetlające obrazki w różnych standardach. Korzystając z przedstawionych danych można również łatwo napisać własny konwerter zmieniający obrazki z jednego standartu na inny. Czekamy na listy z informacjami o waszych osiągnięciach!

JARRI

Copy Party w Gdyni

C-64

W kilka tygodni po C-Party Amigowskim, dnia 21.XII.1991 (sobota) odbyło się w Gdyni party Commodorowskie. O samym zamiarze zorganizowania tej imprezy zostałem poinformowany z dość sporym wyprzedzeniem, jednakże oficjalne zaproszenia zostały rozesłane dopiero tydzień wcześniej, a informacje w magazynach dyskowych na C64 można było znaleźć tylko około dwóch tygodni wcześniej.

Mimo słabej reklamy sporo ludzi zdążyło zgłosić chęć udziału w party, a następnie w ostatnim momencie... odwołać swój przyjazd. Tak więc już same początki nie zapowiadały niczego dobrego. Mimo wszystko - z kilkoma kolegami zdecydowaliśmy się pojechać do Gdyni, aby „zobaczyć jak to będzie".

Podróż rozpoczęła się w zatłoczonym pociągu, który po pięciu godzinach mozolnej jazdy dojechał do Gdyni. Ponieważ przyjechaliśmy już wieczorem 20 grudnia mieliśmy spędzić noc w pobliskim hotelu robotniczym. W dwóch pokojach rozpakowaliśmy nasz cały sprzęt i przystąpiliśmy do pracy, która polegała głównie na katowaniu joystick'ów. W ciągu nocy kilka razy wychodziliśmy na dworzec, aby przyprowadzić na miejsce coraz to nowych gości. O godzinie 9 rano było nas już dziesięciu. Jak było wcześniej ustalone o tej właśnie godzinie miało rozpocząć się party, więc wszyscy zeszliśmy do sali, która była po brzegi wypełniona pustkami. Przenieśliśmy do niej sprzęt i ustawiliśmy wszystko na stołach. Wkrótce też przyszło jeszcze kilka osób i w sumie z wyjątkiem kilku nieznanych ludzi nikt więcej się nie zjawił.

Na całej imprezie obecni byli: z Paradosu: Zak, Sky, Jumbo, Jetboy, Hain, Brush, TG JSL; z Cavernu: Albion, Ignac; z Axel: Alien, Davis, Dirk, Hamster, Meganer, Mikie, Nemo; z Crazy Boys; Crimen, Dexter; z Flatliners: Keen, Liar oraz Polonus z grupy Padua i Sony z grupy Saigon.

Z pewnością potencjalnych chętnych do odwiedzenie party w pewnym stopniu odstraszała cena za wejście - 40 tys. zł. Jest to z pewnością niemało. Z powodu tak małej ilości uczestników organizatorzy byli zmuszeni „opróżnić" salę już o godzinie 16.00.

Podsumowując, muszę niestety stwierdzić, że party się nie powiodło. Organizacja była raczej dobra, zorganizowano porządny bufet, dużą salę oraz pokoje, w których można było się przespać. Zawiedli jednak ludzie... Z pewnością dało się odczuć brak takich grup jak Asphyxia czy Skylight. Coż, miejmy nadzieję że kolejne (być może w Warszawie) copy-party będzie ciekawsze.

JARRI

C-64

Zapraszam do lektury kolejnego odcinka naszego kursu assemblera 6510. Dzisiaj, jak już pisałem miesiąc temu przedstawię rozkazy porównań, przesłań pomiędzy rejestrami oraz odkładania rejestrów na stos. Mam nadzieję, że wszyscy wykonali zadania z poprzedniej części...

ASSEMBLER 6510 - lekcja 7

CMP (compare data and accumlator) - porównaj daną z akumulatorem.

N	V	D	I	Z	C
*	-	-	-	*	*

rozkaz	kod	cykle
CMP #$nn	C9	2
CMP $nn	C5	3
CMP $nn,X	D5	4
CMP $nnnn	CD	4
CMP $nnnn,X	DD	4 (+1)
CMP $nnnn,Y	D9	4 (+1)
CMP ($nn,X)	C1	6
CMP ($nn),Y	D1	5 (+1)

Rozkaz ten odejmuje od akumulatora podaną daną, jednak wynik nie jest nigdzie zapisywany. Natomiast odpowiednie znaczniki są przełączane tak, jak przy wykonywaniu zwykłego odejmowania. Spróbujmy uruchomić program:

```
1000 LDA #$nn
1002 CMP #$80
1004 BRK
```

Zmieniając wartość 'nn' możemy obserwować zmienianie się bitów w rejestrze znaczników.

CPX (compare data and X) - porównaj daną z X

CPY (compare data and Y) - porównaj daną z Y

N	V	D	I	Z	C
*	-	-	-	*	*

rozkaz	kod	cykle
CPX #$nn	E0	2
CPX $nn	E4	3
CPX $nnnn	EC	4
CPY #$nn	C0	2
CPY $nn	C4	3
CPY $nnnn	CC	4

Rozkazy analogiczne do CPX, ale działają na rejestrach X lub Y.

TAX (transfer accumlator into X) - przesłanie akumulatora do X.

TAY (transfer accumlator into Y) - przesłanie akumulatora do Y.

TXA (transfer X into accumlator) - przesłanie X do akumulatora.

TYA (transfer Y into accumlator) - przesłanie Y do akumulatora.

N	V	D	I	Z	C
*	-	-	-	*	-

rozkaz	kod	cykle
TAX	AA	2
TAY	A8	2
TXA	8A	2
TYA	98	2

Te cztery rozkazy służą do przesyłania danych pomiędzy akumulatorem i rejestrami indeksowymi X oraz Y. Rozkazy te są często używane do chwilowego przechowywania danych w innym rejestrze. Takie instrukcje jak TXA czy TYA są też często używane, gdy na jednym z rejestrów indeksowych trzeba wykonać operację dodawania, odejmowania czy też przesunięcia. Wystarczy wtedy tylko przesłać odpowiedni rejestr do akumulatora i wykonać daną operację.

TSX (transfer stack pointer into X) - przesłanie wskaźnika stosu do X.

N	V	D	I	Z	C
*	-	-	-	*	-

rozkaz	kod	cykle
TSX	BA	2

TXS (transfer X into stack pointer) - przesłanie X do wskaźnika stosu.

N	V	D	I	Z	C
-	-	-	-	-	-

rozkaz	kod	cykle
TXS	9A	2

Dwa ostatnie rozkazy są jedynymi, które umożliwiają komunikację ze wskaźnikiem stosu procesora. Przy czym warto zauważyć, iż rozkaz TXS nie wpływa na zawartość rejestru znaczników. Właśnie dzięki tym dwóm rozkazom można w assemblerze 6510 programować proste zabezpieczenia utrudniające analizę kodu (zmiana adresu powrotu z podprogramu). Jednak operacje na stosie należy wykonywać ostrożnie ponieważ podając procesorowi zły adres powrotu można spowodować zawieszenie systemu lub całkiem niekontrolowaną zmianę zawartości niektórych komórek pamięci.

PHA (push accumlator na stack) - umieszczenie akumulatora na stosie.

N	V	D	I	Z	C
-	-	-	-	-	-

rozkaz	kod	cykle
PHA	48	3

Rozkaz ten umieszcza akumulator na stosie procesora oraz zmniejsza wartość wskaźnika stosu o jeden. Podobnie jak TXS i TSX można ten rozkaz używać do zmiany adresu powrotu z podprogramu. Jednak głównym jego zastosowaniem jest chwilowe przechowywanie zawartości rejestrów na stosie. W zestawieniu z rozkazami przesłań można w razie potrzeby zapamiętać cały zestaw rejestrów procesora 6510.

PLA - pull accumlator from stack (pobranie akumulatora ze stosu)

N	V	D	I	Z	C
*	-	-	-	*	-

rozkaz	kod	cykle
PLA	69	4

Rozkaz PLA powoduje zdjęcie ze stosu rejestru akumulatora i zwiększenie o jeden wskaźnika stosu. Zastosowania takie same jak w przypadku PHA.

PHP (push P register on stack) - umieszczenie rejestru znaczników na stosie.

N	V	D	I	Z	C
-	-	-	-	-	-

rozkaz	kod	cykle
PHP	08	3

PLP (pull P register from stack) - pobranie rejestru znaczników ze stosu.

N	V	D	I	Z	C
zawartość pobrana ze stosu					

C-64

rozkaz	kod	cykle
PLP	28	4

Dwa ostatnie rozkazy służą do komunikacji z rejestrem znaczników. Są raczej rzadko używane, chociaż czasem zachodzi potrzeba zapamiętania stanu rejestru znaczników i ponownego ich odtworzenia w dalszej części programu.

Za miesiąc: słów kilka o arytmetyce dziesiętnej (BCD) oraz dokończenie opisu rozkazów procesora.

A oto propozycje rozwiązań zadań z poprzedniego odcinka.

Procedura mnożąca liczbę dwubajtową, która jest zapamiętana w dwóch komórkach na stronie zerowej. Wynik musi być zapisany na trzech bajtach (gdyż wynik mnożenia liczby $FFFF przez $08 jest równy $7FFF8, więc nie mieści się na dwóch bajtach).

```
LDA $FA   ;Przesłanie czynnika do
STA $FC   ;rejestrów, w których ma się
LDA $FB   ;znajdować wynik.
STA $FD   ;
CLC       ;Sekwencja rozkazów mnożąca przez
ROL $FC   ;dwa. Aby pomnożyć daną liczbę przez
ROL $FD   ;osiem należy powtórzyć ją trzy
ROL $FE   ;razy.
```

W przypadku dzielenia wynik możemy zapisać w dwóch bajtach ($FFFF/$08=$1FFF). Dzielenie dwóch bajtów przez dwa będzie wyglądało tak:

```
CLC
ROR $FD
ROR $FC
```

Początek pozostaje taki sam jak w przypadku mnożenia.

Kolej na sumowanie. Oto program:

```
LDA #$00      ;Wyzerowanie rejestrów, w których
STA $FB       ;będzie zapamiętany wynik.
STA $FC       ;
LDX #$00      ;Początek pętli.
TXA           ;Tutaj skaczemy rozkazem BNE.
CLC           ;
ADC $FB       ;Dodanie aktualnej wartości X do
STA $FB       ;komórki pamięci $FB.
LDA $FC       ;Jeżeli wynik większy niż 255 to
ADC #$00      ;dodanie 1 do starszego bajtu.
STA $FC       ;
INX           ;kolejny przebieg pętli.
BNE $petla    ;
RTS           ;KONIEC!!!
```

Trzeci program był chyba najtrudniejszy, jednak ze względu na sporą objętość przedstawię go dopiero za miesiąc.

JARRI



C-64

Programy ciekawe, zwariowane i takie sobie

Paweł Tutka przesłał nam kilka sposobów czyszczenia ekranu.

1. GRAWITACJA.

```
10 FOR A=1 TO 20 : PRINT "$";
20 SYS 59749 : NEXT
30 PRINT" ";
```

Następny program działa dłużej, ale efekt ma też świetny i nazywa się:

2. KURTYNA.

```
10 FOR Y=0 TO 24 : FOR X=0 TO 39
20 POKE 55296+X+40*Y,X/2.5
30 POKE 1024+X+40*Y,160 : NEXT X,Y
40 PRINT " "; : FOR Z=0 TO 50 : ?: NEXT : ?"$";
```

3. OSTATNIA LINIA.

```
10 ? "PAWEL TUTKA"
20 POKE 2051,255 : POKE 2052,255
```

po uruchomieniu i wylistowaniu tego programu, można spróbować zmienić linię o numerze 65536.

Następny program SUPER NEW działa tak, że OLD już jest bezużyteczny.

```
10 A=A+1; POKE A,255 : GOTO 10
```

A teraz premiera czyli **BASIC - SCROLL**

```
10 PRINT" "; : Y=10
20 READ T$
30 FOR X=0 TO 38 : POKE 1024+X+Y*40,
   PEEK (1025+X+Y*40) : NEXT
40 K=K+1:L$=(MID$(T$,K,1))
50 IFL$="£" THEN K=0 : GOTO 20
60 IF L$="↑" THEN RESTORE : K=0 : GOTO 20
70 Z=ASC(L$)-64
80 IF Z<0 THEN Z=Z+64
90 POKE 1063+Y*40,Z : GOTO 30
100 DATA GIGABAJTOWE POZDROWIENIA DLA
    REDAKCJI £
101 DATA 64PLUS4 & AMIGA £
102 DATA PRZESYLA COMMONESIS ↑
```

Każda linia data powinna kończyć się znakiem £, a ostatnia ↑ oznaczając, że Scroll ma być wyświetlany od początku. Zaletą tego scroll'a jest to, że możemy umieścić długi napis (do ostatniej wolnej komórki).

Ostatni program jest szczególnie dla tych osób, które lubią korzystać z przełączania bloków programu za pomocą spacji.

```
10 FOR A=4096 TO 4112 : READ W : POKE A,W : NEXT
20 ?"PRESS SPACE"
30 SYS 4096 : GOTO 20
40 DATA 173, 32, 208, 105, 1, 141, 32, 208
50 DATA 173, 1, 220, 201, 239, 208, 241, 96,
```

Aby otrzymać kolorowe DIRectory należy zmienić nazwę dysku, tak aby np.

```
Z$=CHR$(34)
T$="tytul max 14 znakow z uwzglednieniem kolorow"
SAVE Z$+T$+Z$,8
```

W zmiennej T$ możemy zmieniać kolor, włączać i wyłączać negatyw, czyścić ekran itp.[...].

Następne programy przesłał nam Maciek Kędzierski.

Program zatytułowany jest „Tłuste litery" i powoduje, że standardowy generator jest modyfikowany i przepisywany w inny obszar.

Oto program assemblerowy:

```
        *=C000
        LDA  #00
        STA  FB
        STA  FD
        LDA  #D0
        STA  FC
        LDA  #E0
        STA  FE
        LDX  #10
LOOP    LDY  #00
LOOP1   LDA  (FB),Y
        LSR
        ORA  (FB),Y
        STA  (FD),Y
        DEY
        BNE  LOOP1
        INC  FC
        INC  FE
        DEX
        BNE  LOOP
        RTS
```

Listing w Basic'u:

```
10 FOR I=1 TO 36 : READ A : POKE 49151+I,A : NEXT
20 POKE 56334,0 : POKE 1,51
30 SYS 49152
40 POKE 1,55 : POKE56334,1
50 POKE 648,204 : POKE 53272,58
60 POKE 56576, PEEK(56576) AND 252
70 END
100 DATA 169,0,133,251,133,253,169,208,133,252,169,
    224,133,254,162,16,160,0
110 DATA 177,251,74,17,251,145,253,136,208,246,
    230,252,230,254,202,208,237,96
```

Drugi z moich programów jest napisany w Basic'u i powoduje, że dowolne teksty są drukowane po literce.

„Drukowanie po literce"

```
5 DIM A$(255)
10 PRINT CHR$(147)
20 A$= "DRUKOWANIE PO LITERCE 64 PLUS 4
   & AMIGA"
30 FOR A=1 TO LEN(A$)
40 FOR I=0 TO 40 : NEXT I
50 PRINT MID$ (A$,A,1); : NEXT A
```

```
60  PRINT : PRINT : GOTO 30
```

Pętla w linii 40 powoduje opóźnienie szybkości drukowania.

Kolejny program napisany jest również w Basic'u i jest ulepszeniem programu Jacka Zaczko nr 11/91. Drukuje on cały ciąg, a nie pojedyncze wyrazy. Nie czyści całego ekranu, a jedynie ustawia kursor w lewym, górnym rogu, za każdym razem zwiększa o jeden kolor, co daje efekt migotania napisu.

```
10  DIM A$(255) : PRINT CHR$(147)
15  B$="COMMODORE 64+4 & AMIGA' PREZE"
20  C$="NTUJE: PLYNNY POZIOMY PRZESOW
    TEKSTOW + MULTICOLOR!!! COPYRIGHT (C)"
25  D$="1992!!! AUTOR PROGRAMU: MACIEJ
    KEDZIERSKI"
30  A$=B$+C$+D$
35  FOR A=1 TO LEN(A$) : POKE 646,A
40  PRINT CHR$(19) ; MID$(A$,A,40);
45  FOR I=0 TO 50 : NEXT I
50  NEXT A
55  GOTO 35
```

Ważne jest aby 40 pierwszych znaków ciągu A$ były spacje. Spacją musi być także ostatni znak. Pętla w linii 45 określa opóźnienie czyli szybkość przesuwu[...].

Kolejny program przesłał do nas Maciej Rzemieniecki, oto listing tego programu:

```
10  FOR A=0 TO 100
20  POKE 53280,A
30  NEXT
40  POKE 53281,A
50  GET A$ : IF A$ : " " THEN POKE 53281,8 : END
60  GOTO 40
```

Autorom dziękujemy za nadesłane programy. Pozostałych Czytelników zapraszamy do wspólnego redagowania tej rubryki.

W.S.

SPIS ZESTAWU PUBLIC DOMAIN PACK NR 13

C-64

(styczeń '92)

W styczniowym (13) zestawie PDP, oprócz kilku ładnie zrobionych programów demonstracyjnych, umieściliśmy najnowsze numery trzech polskich magazynów. Oprócz tego znalazło się też na nim kilka przydatnych programów użytkowych. Prezentujemy poniżej ich krótkie opisy.

1. Char Zoomer V3.1. Przy pomocy tego programu można każdą czcionkę wielkości 8 na 8 pikseli powiększyć do rozmiarów 16 na 8, 8 na 16 lub 16 na 16 pikseli. Jest to przydatne głównie wtedy, kiedy mając do dyspozycji jedynie małą czcionkę chcemy napisać program przesuwający napisy o większych rozmiarach.



2. Colour Bar Editor v3.0. Program ten służy do łatwego ustawiania tablicy kolorów dla procedury tworzącej na ekranie poziome, kolorowe pasy (tzw. colour bars).

3. Hires + A-FLI Editor. Jest to połączenie dwóch niemalże identycznych programów graficznych. Pierwszy z nich to edytor do zwykłego trybu graficznego Hi-Res. Tak więc przy wymiarach obrazu 320 na 200 pikseli, w każdym polu o wielkości 8 na 8 pikseli można użyć dwóch kolorów (tło+atrament). Drugi program umożliwia tworzenie grafiki w technice A-FLI. Dzięki niej możliwe jest użycie dwóch kolorów (również tło+atament) w każdym polu o wielkości 8 na 1 piseli przy całkowitej wielkości ekranu 308 na 200 pikseli. Niestety trzy pierwsze kolumny ekranu nie mogą być używane przez grafikę (w tej rozdzielczości). Drugą wadą trybu A-FLI jest duża ilość danych potrzebnych na opisanie obrazka (około 16kB) oraz konieczność zastosowania specjalego programu wyświetlającego.

4. FLI Designer v1.1. Jest to chyba najlepszy edytor do tworzenia grafiki w technice FLI. Podobnie jak w technice A-FLI tak i tu obrazki zajmują dużo pamięci i potrzebna jest specjalna procedura wyświetlająca. Jednakże w zamian za te niedogodności możliwe jest użycie przy jednym, ogólnym kolorze tła, trzech kolorów atramentu w każdym polu wielkości 4 na 1 pikseli przy rozdzielczości 148 na 200. W zasadzie znikają niemalże wszelkie ograniczenia w użyciu kolorów (o technice FLI oraz A-FLI napiszemy wkrótce na łamach C64+4 nieco więcej).

5. Dirmaster v3.1. Program ten to bardzo wygodny edytor katalogu dyskietki. Dzięki niemu łatwo można zmieniać nazwę, długość i typ dowolnego pliku, nazwę, identyfikator oraz ilość bloków wolnych dyskietki. Można także zmieniać kolejność plików w katalogu oraz dostawiać tzw. „puste" pliki. Oprócz bardzo dużej ilości opcji edycyjnych program posiada

C-64

również możliwość szybkiego formatowania (9 sek.) i „odśmiecania" (15 sek) dysku.

5. **Accesinus**. Program ten służy do tworzenia niemalże dowolnego ruchu sprite'ów po ekranie. Dzięki bardzo dużej ilości ustawianych parametrów można stworzyć sporo ciekawych efektów. Po ustawieniu parametrów wystarczy zgrać całość na dysk. Korzystać z gotowej procedury można w Basic'u, gdyż korzysta ona jedynie z przerwań IRQ.

7. **Music Routine Cruncher v1.5**. Jest to najnowsza wersja programu służącego do skracania czasu odtwarzania muzyczki. Program ten przydatny jest raczej bardziej zaawansowanym koderom.

8. **Gandalf Protector v3.0** - zabezpieczanie swoich programów przed łatwym dostępem osób nieporządanych.

9. **Gandalf Coder v1.0**. Przy pomocy tego programu można zabezpieczyć swoje prace „na hasło", o którego podanie przy późniejszym uruchamlaniu będzie prosił komputer.

10. **Disk-tape copy v2.0**. Jest to bardzo starannie wykonany program kopiujący pliki z dysku na taśmę.

11. **Gnd-packer v1.0**. Program służący do pakowania plików.

Mamy nadzieję, że programy te przydadzą się w pracy szerokiemu gronu naszych czytelników.

JARRI

W co pogramy?

W ciągu kilku minionych tygodni wiele firm softwa-re'owych, licząc na większą sprzedaż w okresie świątecznym, zaprezentowało swoje nowe produkty. W tym artykule przedstawię kilka gier, które warto dołączyć do swojej biblioteki programów na C64.

Przeważająca część ostatnio wydanych gier to tzw. zręcznościowe. Do takich należy FINAL FIGHT firmy U.S. Gold (karate). Gra jest bardzo podobna do starej serii RENEGADE i można ją uznać za jej kontynuację. Warto również wymienić grę DARKMAN (Ocean), która z kolei przypomina dawny przebój THE UNTOUCHABLES.

Firma U.S. Gold zaprezentowała również dość oryginalną grę zręcznościową ALIEN STORM, w której naszymi przeciwnikami są dziwne stwory.

Z pewnością miłośników „bijatyk" ucieszy ukazanie się zręcznościowej wersji gry TEENAGE MUTANT HERO TURLES (firma Konami). Gra ma dobrze dopracowaną oprawę zarówno graficzną jak i dźwiękową, a przez niektórych została uznana najlepszą grą „beat 'em up" roku 1991.

W końcu można też już zagrać w długo oczekiwaną grę TERMINATOR II, stworzoną na podstawie filmu. Jest w niej prawie wszystko: jazda na motocyklu, ucieczka ze szpitala, walka T1000 z T800. W sekwencje typowo zręcznościowe wplecione są fragmenty, w których należy ułożyć, w zadanym czasie, dość prostą układankę.

Zwolenników ostrej walki ucieszą zapewne dwie gry symulujące wrestling. Pierwsza z nich to PIT FIGHTER z firmy Domark. Walczymy na ulicy z kolejnymi, coraz to trudniejszymi do pokonania, przeciwnikami. Do dyspozycji mamy wiele ciosów, z których ciekawsze to... rzucenie przeciwnikiem o podłogę. Gra różni się od innych tym, że nawet gdy jeden z graczy został powalony walka nie zostaje przerwana.

Druga gra wrestlingowa to WRESTLEMANIA z firmy OCEAN. Jest bardzo do starannie zrobiona pod wzgiędem grafiki i dźwięku.

Ostatnio na rynku pojawia się coraz mniej dobrze zrobionych „strzelanin". Jedyna jaka pojawiła się w ostatnim czasie to SUPER SPACE INVADERS z firmy Domark. Jest to kolejna konwersja bardzo już starej gry SPACE INVADERS polegającej na zestrzeleniu wszystkiego, co rusza się w górnej części ekranu. Jedyną zaletą programu jest całkiem niezła grafika.

Warto jeszcze odnotować pojawienie się zręcznościowej wersji gry ELVIRA (firma Flair Software). Niestety nie jest ona zrobiona starannie ani pod względem graficznym, ani dźwiękowym.

Także miłośnicy wyścigów samochodowych znajdą dla siebie kilka dobrych pozycji. Ukazała się min. zapowiadana gra TURBO CHARGE firmy System 3. Jest ona podobna do starego hitu OUTRUN, jednak z dużo lepszą grafiką i muzyką. Również zadanie uległo pewnej zmianie: nasz samochód wyposażony jest w karabin, a my gonimy groźnych przestępców.

Druga gra jest już stuprocentową kontynuacją OUTRUN'a, a nosi tytuł OUTRUN EUROPE. Oprócz zwykłej jazdy samochodem, autorzy wprowadzili również wyścigi na motocyklu oraz na ślizgaczu.

W końcu została również zrobiona wersja na C64 znanego hitu z Amigi - LOTUS ESPRIT TURBO CHALLENGE firmy Gremlin. Niestety autorzy konwersji nie wykorzystali pełnych możliwości C64 i gra prezentuje się raczej ubogo, głównie pod względem oprawy muzycznej.

Również niektóre mniejsze firmy wypuściły na rynek swoje nowe produkty. Wszyskich miłośników sympatycznego jajka - DIZZY'iego z pewnością ucieszy ukazanie się dwóch nowych: SPELLBOUND DIZZY oraz YOLK DIZZY.

Również miłośnicy strzelanin otrzymalją dwie gry niemal idealnie dobpracowane pod względem grafiki i muzyki. Pierwsza to typowa strzelanina pod tytułem RUBICON, druga to gra łącząca wątek przygodowy ze zręcznościowym - CLYSTRON.

W końcu możemy obejrzeć gry pisane przez znaną grupę COSMOS DESIGNS. NIBBLY to świetna gra zręcznościowa, druga - PLURAL - to na pozór zwykła strzelanina, wyróżniająca się jednak dużą ilością obiektów oraz różnorodnością rodzajów broni.

Jak widać, wszyscy maniacy joystick'a znajdą dla siebie jakiś ciekawy program na najbliższe kilka tygodni.

JARRI

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C - 64

Styczeń '91 (nr 1)

STRONA A

- Mega demo grupy „VISION"-MIST2

STRONA B

- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

Luty '91 (nr 2)

STRONA A

- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+(DEMO)[02]
- DREPTACZ _ BASIC

STRONA B

- SWISS CHEESE/CFA
- DISK FAST LOADER

Marzec '91 (nr 3)

STRONA A

- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELAKATOR 64
- KOREKTOR 64
- FLASH

STRONA B

- HOT SHOT nr9 (zach. magazyn fanów)
- BAD NEWS nr2 - j.w.
- DEMO - rekord - 290 SPRITE'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER _ adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

Kwiecień '91 (nr 4)

STRONA A

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku

STRONA B

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika

Maj '91 (nr 5)

STRONA A

- CRUEL SOLIDERS - demo
- DESTINATION - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

STRONA B

- MEGA DEMO „INFOSYSTEM 91"

Czerwiec '91 (nr 6)

STRONA A

- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

STRONA B

- PSC - MAG #9'06/91
- CONSPIRE?OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

Lipiec '91 (nr 7)

STRONA A

- Mega demo „MY, OH MY!" grupy LIGHT

STRONA B

- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

Sierpień '91 (nr 8)

STRONA A

- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dismaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

STRONA B

- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

Wrzesień '91 (nr 9)

STRONA A:

- Mega Demo grupy FLASH

STRONA B:

- Hot Shot - magazyn dyskowy
- Code Sucker monitor - pr. użytkowy grupy PADUA
- Mountain Ride - gra w BASIC

Październik '91 (nr 10)

STRONA A i B

- MEGA DEMO „AIRDANCE 4" grupy T.A.T.

Listopad '91 (nr 11)

STRONA A

- NEW LAW & ORDER!
- FLT/LEGOLAND
- FLT/LEGONOTE
- TERMINAT.2%/FLT

STRONA B

- SM. CRIMINAL #08
- SMALL BUT FINE
- HOLLY SMOKE/M12
- UNITE!/SYLVIO

Grudzień '91 (nr 12)

STRONA A

- ARMAGEDDON 5
- NOTE TO DEMO

STRONA B

- OUTRUN 2 MUS $ SFX
- AFTERBURNER/MON
- TRIVIA-GAME MUSIC
- FORM.I. SIMULATOR
- 2400AD END-TUNE
- NIGHTHUNTER DIGI
- ELIMINATOR MUSIC
- TOMCAT MUS./MON
- ZAMZARA TUNE/MON
- NOTE TO DISK
- HIGHLIFE#9

PUBLIC DOMAIN PACK AMIGA

Styczeń '91 (nr 1)

- Programy kompresorów danych
- Grafiki Borysa Vallejo
- Prezentacja najlepszych muzyczek
- INTUITRACKER

Luty '91 (nr 2)

- Request player; Multi ripper
- 3-rd day; Phantasmagoria - demo
- Master Seka; Virus Ekspert v1.6
- AMOS-programy; Moduły: Killing game show, Upon Me, Let's swing it.

Marzec '91 (nr 3)

- Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)
- Najlepsze muzyczki: NOW WAIT? - DR.AWESOME
- AMOS - procedury
- DEMO grupy REBELES „TOTAL TRIPLE TROUBLE"

Kwiecień '91 (nr 4)

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  - > TECHNOSTYLE 2
  - > GALAXY 2
- GRAFIKA - prezentujemy rysunki
  - > RICK PARKS

Maj '91 (nr 5)

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  - > MIAMI VOICE
  - > ANTI ATARI SONG

Czerwiec '91 (nr 6)

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

Lipiec '91 (nr 7)

- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

Sierpień '91 (nr 8)

- Real 3D - demo nowego programu do raytraceing'u
- Moduł Muzyczny XTC STEREO

Wrzesień '91 (nr 9)

- MODUŁY MUZYCZNE dla programu TFMX:
  - > R - TYPE
  - > THE HOUSE OF TECHNO
- VIRUS EXPERT v181 + 143
- BOOT BLOCK'I
  - > BOOTX v 3.80
  - > IMPLODER v 4.0

Październik '91 (nr 10)

- ANARCHY - „THE INSPIRATION IS NONE"
- DUAL CREW - „NEW DIMENSION"
- SANITY - „ELYSIUM"

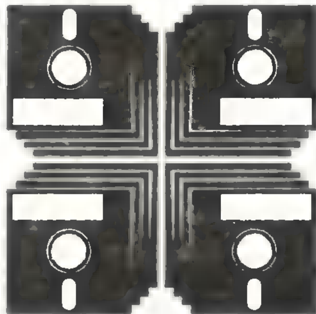
PUBLIC DOMAIN PACK

Listopad '91 (nr 11)

- COMPUTER HEAD - animacja
- MODUŁY POD MEDPLAYER
  -CONFUSED
  -ROCKED
  I WIELE SAMPLI
- SAVE GAME „MONKEY ISLAND"

Grudzień '91 (nr 12)

- GEM X
- BOOTX V. 4.13
- FINAL KIT -monitor
- MEGA-MON
- VARIA



PUBLIC DOMAIN PACK C-64
TAPE NR 1

- TURBO
- SINUSDATA - EDITOR
- FAST CRUNCHER V3
- ANAL S.C. IBEYOND
- VECTOR - VICTORY
- PUZZLENOID+4
- TUNE OF MONTH #1
- NIM
- STRZAŁKA 64+
- LOGO - WRITER V.2.0
- CAN'T TOUCH IKU!
- NTRO PRV
- BONZIEED!!
- ZAX PACKIS
- READ THIS FIRST
- COMMERCIAL BREAK
- 290 SPRITES!
- NOTE - ABOUT
- BAD NEWS NR2
- TO BAD NEWS...
- CONTACT CORNER!
- PROJEKT DUSZKÓW
- SYMPHONY NR14
- SYMPHONY NR15
- SYMPHONY NR16
- SYMPHONY NR17
- SYMPHONY NR18
- SYMPHONY NR19
- CRUISER/GIANTS
- NOTE>ANO<PADUA
- LET'S DYSP!
- FINALTAPE
- MUSIC - SEARCHER

PUBLIC DOMAIN PACK C-64
TAPE NR 2

- TURBO
- PUBL. DOMAIN. INFO
- FONTGRUB 1.0
- DREPTACZ BASIC
- LOAD DIS FIRSY
- MACROASSEMBLER
- TURBOASSEMBLER
- RELOCATOR
- LOGOPAINTER 3!
- REASSEBLER
- SPRITE - EDITOR
- FAST - CRUEL U.2.5
- HIGHLIFE NR5
- AXEL NEWS NR1
- GWIAZDY
- FLIGRAPH 2.2/BML
- NOTE TO FLI V.2.2
- DISKNOTKA/PADUA
- MEGA PACKER/T
- MIST II/ VISION
- TTECHSCR &DYSP
- PLASMA - WORLD
- VECTORBOBS...
- VECTOR - PLOTS
- FLI - UPSCROLL
- BORDER - HIRES
- ROCK AROUND
- FACEWRITER
- CHAR EDIT 2+2
- DISKNOTER
- DESTINATION'91
- CONTACTDEMO/ORE
- FONTEDITOR
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 3

- TURBO
- PUBLIC DOMAIN NOTE
- GRAVEYARD NOTES!
- NOTE FROM BEAT!!
- ANONYM SPEAKING!
- SNDK. V3.7/TOPAZ
- AFLI - EDITOR
- NOTER V2.2/TOPAZ
- DLW V1.5/TOPAZ
- CODE - S.MON/PADUA
- OPINION - POLL/PDA
- MOUNTAIN RAID
- PART 1
- PART 2
- PART 3
- PART 4
- PART 5
- FAIRLIGHT 1
- FAIRLIGHT 2
- FAIRLIGHT 3
- FAIRLIGHT 4
- FAIRLIGHT 5
- THE END

PUBLIC DOMAIN PACK C-64
TAPE NR 4

- TURBO
- OUT RUN 2 MUS & SFX
- AFTER BURNER/MON
- FORM.1.SIMULATOR
- 2400 AD.END - TUNE
- NIGHT HUNTER DIGI
- ELEMINATOR MUSIC
- TOMCAT MUSIX/MON
- ZAMZARA TUNE
- DYNAMIX TUNE
- HIGHLIFE NR9
- SNAKES C3
- SNARK C3
- SNERD C3
- WAREHOUSE C3
- STARTREK C3
- TOWER
- SNOOPY
- NEW LAW & ORDER
- FLT/LEGONOTE...
- TERMINAT.2%/FLT
- UNITE!/SYLVIO
- BALL - SCOPE/451
- TRIVIA - GAME MUS.
- RESET - MONITOR
- HOLY SMOKE

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr: 5.09011-400522.7-2511-30-111.0 następujące kwoty: 20.000zł za pojedyńczy zestaw dyskowy dla C-64, 30.000 zł za zestaw programów PD na kasecie, 25.000zł za zestaw dla Amigi.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64D"** - jeśli zamawiamy zestaw dla C-64 na dyskietce lub **„PDP-64T"** - dla zestawu taśmowego, zestaw dla Amigi prosimy zaznaczać skrótem **„PDP-A"** - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty.

W prenumeracie zestawy kosztują: **PDP-64 - 18.000zł** (12 numerów 216 tys. zł), **PDP-A - 22.000 zł** (12 numerów 264 tys. zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Powyższe warunki odnoszą się również do naszych zestawów wydanych w 1991r.

Zestawy taśmowe PDP-64 w 1992r. będą ukazywały się w miarę napływu nowych, ciekawych programów - o czym będziemy informować na łamach naszego pisma.

Zamów nie zwlekaj!

C-64 VOICETRACKER V4.0

Rewelacyjny program muzyczny!



Tylko [redacted] kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową [redacted] programem VOICETRACKER V4.0, instrukcję obsługi, oraz - dodatkowo - przykładowe demonstracje muzyczne. **UWAGA! Wersja magneto-[redacted] tylko 40.000 [redacted]**

Przedsiębiorstwo ABUK posiada wyłączność [redacted] dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach oraz wymiany dyskietki [redacted] razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej 10 kompletów udzielamy [redacted] rabatu).
Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000 zł (wersja dyskowa) lub 40.000 zł (taśma) [redacted] konto: Bank PKO SA Bydgoszcz, konto [redacted] 5.09011-400522.7-136-11-111.0. Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres [redacted] z dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

W związku [redacted] pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi [redacted] nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet [redacted] może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzedaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA

Odcinek dla Poczty

Zł ....

słownie ....

....

wpłacający ....

....

....

(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata

zł..........

Odcinek dla posiadacza rachunku

Zł ....

słownie ....

....

wpłacający ....

....

....

(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata

zł..........

Odcinek dla Banku

Zł ....

słownie ....

....

wpłacający ....

....

....

(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata

zł..........

Odcinek dla wpłacającego

Zł ....

słownie ....

....

wpłacający ....

....

....

(dokładny i CZYTELNY adres)

na rachunek:
Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.

Opłata

zł..........



TREŚĆ ZAMÓWIENIA:

Prosimy ▫ CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy ▫ CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Prosimy ▫ CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA:

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-64 i Amiga. Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane). Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu. Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po

500.000 zł każda.

Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma. Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA"

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW OD LISTOPADA 199OR. DO GRUDNIA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES :
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.
(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

W dzisiejszym kąciku zajmiemy się pisaniem pierwszych programów. Na początku zajmiemy się procedurami wykorzystywanymi w demach (i nie tylko!).

AMIGA

KĄCIK POCZĄTKUJĄCEGO KODERA CZ.10

1. Czekanie na „ramkę".

Jako pierwszą przedstawiam podstawową procedurę, która oczekuje na określoną linię ekranu (czyli linię, którą aktualnie rysuje monitor za pomocą działa elektronowego na fluorescencyjnej powierzchni kineskopu). Jeżeli piszemy procedurę, która ma na przykład przesuwać (scroll) część ekranu zawartą pomiędzy liniami $80 a $ff, to należy ją wykonywać gdy działo elektronowe znajduje się w innym obszarze (powyżej lub poniżej zadanego obszaru), gdyż w przeciwnym przypadku (np. gdy będziemy wykonywać procedurę w linii $c0) nasz scroll będzie złamany. Dzieje się tak dlatego, iż ekran zostaje wykreślony do linii $c0, a następnie, po wykonaniu procedury scrolla, część wykreślona zostaje przesunięta i nasz scroll będzie wydawał się złamany (dolna część będzie przesunięta względem górnej o pewien fragment - ale tak wygląda to tylko na ekranie).

Najlepiej jest oczekiwać na ostatnią linię ekranu, gdyż w czasie gdy monitor cofa działo z ostatniej linii do pierwszej mamy bardzo dużo czasu „ekranowego" zwanego „ciemną fazą".

Za numer aktualnie kreślonej linii odpowiadją komórki VPOS i VHPOS, a numer aktualnie kreślonej linii określają bity 15-8 z komórki VHPOS oraz bit 0 z komórki VPOS.

A teraz już przykładowa procedura (a właściwie procedury).

Przykład 1.1 (czekanie na zadaną linię)

```
WaitRaster:
      move.l$dff004,d0
      lsr.l#8,d0
      and.w#$1ff,d0
      cmp.w#$12d,d0
      bne.sWaitRaster
rts
```

Linia 1 - Przepisanie zawartości komórek VPOS (-adres $DFF004) oraz VHPOS (adres $DFF006) do rejestru D0.

Linia 2 - Przesunięcie w prawo zawartości rejestru d0 (aby bity 16-8 znalazły się na pozycjach 8-0).

Linia 3 - Logiczne "i" aby pozostawić tylko bity 8-0 w rejestrze D0 a pozostałe wyzerować.

Linia 4 - Porównanie z numerem linii na którą chcemy czekać (linia $12d to linia kończąca ekran - niżej w zasadzie nic się nie robi).

Linia 5 - W przypadku gdy jeszcze nie monitor nie osiągnął linii $12d skok do początku procedury.

Linia 6 - Powrót z procedury Wait Raster (gdy jesteśmy w danej linii).

Przykład 1.2 (czekanie na zadaną linię o numerach od $2c do $ff).

```
WaitRaster:
      cmp.b#$80,$dff006
bne.sWaitRaster
rts
```

Linia 1 - Porównanie komórki $dff006 z numerem linii na którą chcemy czekać.

Linia 2 - W przypadku gdy jeszcze nie osiągnięto tej linii skok do WaitRaster.

Linia 3 - Powrót z procedury

Przykład 1.3 (oczekiwanie na zadaną linię w zakresie $100 - $12d).

```
WaitRaster:
      cmp.b#$ff,$dff006
bne.sWaitRaster
WR2:  cmp.b#$2d,$dff006
bne.sWR2
rts
```

Linia 1 - Porównanie komórki $dff006 z wartością $ff (ostatnia linia górnej części ekranu).

Linia 2 - Powtarzanie porównania dopóki linia nie zostanie osiągnięta.

Linia 3 - Porownanie z zadanym numerem linii (od $00 do $2d).

Linia 4 - Powtarzanie porównania.

Linia 5 - Powrót gdy dana linia została osiągnięta.

Przykład 1.4 (oczekiwanie na zadaną linię w zakresie $100 - $12d)

```
WaitRaster:
      btst#0,$dff005
bne.sWaitRaster
```

```
WR2:    cmp.b#$2d,$dff006
        bne.sWR2
        rts
```

Linia 1 - Test najstarszego bitu numeru linii, który jest reprezentowany w komórce VPOS jako bit najmłodszy.

Linia 2 - Powtarzanie porównania dopóki nie zostanie osiągnięta dolna część ekranu.

Linia 3 - Porównanie z zadanym numerem linii (od $00 do $2d).

Linia 4 - Powtarzanie porównania.

Linia 5 - Powrót, gdy dana linia została osiągnięta.

Myślę, że te cztery przykłady wystarczą aby zrozumieć metody oczekiwania na odpowiednią linię.

2. Rozpakowywanie obrazka w IFF'ie.

IFF czyli Interchange File Format to format zapisu danych, aby ułatwić ich przenoszenie pomiędzy programami. Standardowi temu poświęciliśmy w "64+4" artykuł a dzisiaj w Kąciku Początkującego Kodera zajmiemy się procedurą służącą do rozpakowywania danych rysunku. Procedura ta oprócz dekompresji danych BODY wystawi jeszcze pozostałe dane.

```
*********************************************
*   Vision Thing Software Group             *
*                                           *
* Project:  IFF Depack Routine              *
* Coded:    Marcin "Duddie" Dudar           *
* Version:                                  *
*        v1.0 - Jan 23, 1991                *
*********************************************

IFF:      equ $40000     ; adres danych w IFF'ie
Picture:  equu $60000    ; adres wolnego obszaru
                           pamięci gdzie zostanie
                           rozpakowany rysunek.
IFFtoRAW:
          movem.l  d1-d7/a0-a6,-(sp)   ; rejestry na stos
          lea   IFF,a5           ; adres obrazka w IFF'ie
          cmp.l    #'FORM',(a5)+     ; sprawdzenie
                                       czy jest to IFF
          bne Error          ; jeżeli nie to błąd
          addq.l   #4,a5
          cmp.l    #'ILBM',(a5)+     ; sprawdzenie czy jest
                                       to rysunek w formacie
                                       IFF (InterLeaved BitMap)
          bne Error          ; jeżeli nie to błąd
Loop:
          move.l   (a5)+,d1          ; wpisanie typu
                                       Chunk'a do D1
          move.l   (a5)+,d2          ; długość Chunk'a do D2
          cmp.l    #'BMHD',d1        ; czy jest to BMHD
                                       (BitMap HeaDer)
          beq.s    BMHD  ;           jeżeli tak to skok do
                                       procedury BMHD
          cmp.l    #'CAMG',d1        ; czy jest to CAMG
                                       (Commodore AMiGa)
                                       - Chunk o danych
                                       specjalnie dla Amigi
                                       (HAM, etc.)
          beq CAMG         ; jeżeli tak to skok do CAMG
          cmp.l    #'BODY',d1      ; czy jest to Chunk
                                BODY (Bitplanes and Optional
                                mask interleaveD bY row)
          beq.s    BODY  ; jeżeli tak to skok do BODY
LoopEnd:
          add.l    d2,a5   ; dodanie długości Chunka do
                             adresu w strukturze IFF (inne
                             Chunki niż BODY, BMHD i CAMG
                             są ignorowane
          bra.s    Loop    ; skok do pętli głównej
TheEnd:                    ; wszystko OK
          movem.l  (sp)+,d1-d7/a0-a6  ; rejestry ze stosu
          moveq    #0,d0              ; OK do D0
          rts                         ; powrót
Error:
          movem.l  (sp)+,d1-d7/a0-a6  ; rejestry ze stosu
          moveq    #-1,d0             ; błąd do D0
          rts              ; powrót z procedury
BODY:                      ; rozpakowanie danych
          move.l   a5,a0   ; do A0 wskaźnik danych
          lea  Picture,a1  ; do A1 miejsce gdzie dane
                             zostaną rozpakowane
          moveq    #0,d0   ; 0 do D0 aby wyczyścić cały rejestr
          move.b   Depth,d0         ; ilość bitplane'ów do D0
          move.w   BytesPerRow,d1   ; szerokość
                                      obrazka w bajtach
          move.w   Height,d2        ; wysokośc obrazka
          bsr ul_DepackIFF   ;wywołanie procedury
                              rozpakowania obrazka
          bra.s    TheEnd  ; skok do końca
BMHD:                      ; odczytanie danych o obrazku
          move.w   (a5),Width       ; szerokość obrazka do
                                      komórki Width
          move.w   2(a5),Height     ; wysokość obrazka do
                                      komórki Height
          move.b   8(a5),Depth      ; ilość bitplane'ów do
                                      komórki Depth
          move.w   (a5),d1          ; szerokość w pikslach
                                      do D1
          move.w   d1,d3   ; D1 do D3 (szerokość)
          and.w    #$7,d3           ; pozostawienie
                                      najmłodszych trzech
                                      bitów szerokości
          lsr.w    #3,d3   ; przesunięcie w prawo o trzy
                             czyli dzielenie liczby przez osiem
                             aby uzyskać szerokość obrazka
                             w bajtach
          tst.w    d3      ; test końcówki (jeżeli jest różna
                             od zera to oznacza, że biplane
                             jest szerszy o jeden bajt)
          beq.s    He      ; skok gdy D3 jest równe 0
          addq.w   #1,d1   ; zwiększenie szerokości bitplane'u
                             o jeden
He:
          btst #0,d1       ; test najmłodszego bitu szerokości
                             (biplane musi być szeroki na
                             parzystą ilość bajtów)
          beq.s    He2     ; skok gdy parzysta ilość
                             (bit wyzerowany)
          addq.w   #1,d1   ; zwiększenie szerokości aby
                             wyrównać do parzystej
He2:
          move.w   d1,BytesPerRow   ; szerokość do
                                      komórki BytesPerRow
          bra LoopEnd      ; skok do pętli głównej
CAMG:
          move.w   (a5),PadWord     ; dane dla formatu IFF
                                      specyficzne dla Amigi
          move.w   2(a5),ViewModes
          bra LoopEnd      .; skok do pętli głównej
; Depack Byte1Run Compressed Body ; v1.0 Finished on
                                    Nov 10, 1991, 11:25 PM
ul_DepackIFF:
          movem.l  d0-d7/a0-a6,-(sp)   ; rejestry na stos
          and.l    #$fff,d1         ; pozostawienie 12 bitów
                                      z szerokości
          and.l    #$fff,d2
          move.l   d1,d3            ; szerokość do D3
```

```
        mulu     d2,d3  ; pomnożenie D3 przez D2 czyli
                          szerokości przez wysokość aby
                          otrzymać wielkość jednego
                          bitplane'u
        move.l   a1,a2  ; adres pamięci gdzie dane będą
                          rozpakowywane
        subq.l   #1,d2  ; zmniejszenie D2 o jeden (główna
                          pętla ma być wykonana tyle razy
                          jaka jest wysokość, ale że petla
                          DBcc - wykonuje się x razy
                          plus 1 dlatego należy zmniejszyć D2)
        subq.l   #1,d0  ; zmniejszenie D0 zawierającego
                          ilość bitplane'ów (także dla pętli
                          tyle, że wewnętrznej, która będzie
                          wykonywana dla każdej linii)
U2_Get:
        move.l   d0,d6  ; licznik pętli wewnętrznej do D6
        move.l   a2,a3  ; adres linii bitplane'u
U2_NextPlane:
        move.l   a3,a1  ; adres bitplane'u
        moveq    #0,d7  ; wyzerowanie D7 - zawiera ilość
                          bajtów zdekompresowanych w linii
U2_TakeNext:
        moveq    #0,d5  ; wyzerowanie D5
        move.b   (a0)+,d7        ; pobranie bajtu
                          skompresowanego
        bmi.s    U2_Xero         ; jeżeli ujemny to skok
                                   do procedury
                                   powielające j następny
                                   bajt
U2_Copy:
        move.b   (a0)+,(a1)+     ; kopia bajtu jeżeli nie
                                   jest skompresowany
        addq.l   #1,d7  ; zwiększenie licznika szerokości
        dbf d5,U2_Copy  ; pętla kopiująca ilość
                          bajtów zawartą w D5
U2_Check:
        cmp.w    d1,d7  ; czy koniec linii
        bcs.s    U2_TakeNext     ; jeżeli nie to skok do
                                   początku pobierania
                                   kolejnego bajtu
        add.l    d3,a3  ; zwiększenie A3 o wielkość jednego
                          bitplane'u aby wskazywał następny
        dbf d6,U2_NextPlane      ; pętla dla ilości
                                   bitplane'ów
        add.l    d1,a2  ; zwiększenie A2 o jedną linię
        dbf d2,U2_Get   ; pętla dla ilości linii
        movem.l  (sp)+,d0-d7/a0-a6  ; rejestry ze stosu
        rts             ; powrót
U2_Xero:
        move.b   (a0)+,d4        ; pobierz bajt do
                                   powielania
        neg.b    d5     ; negacja D5 (ilość bajtów jest
                          zmniejszona o jeden i zanegowana
PowielajPetla:
        move.b   d4,(a1)+        ; powielanie bajtu
        addq.l   #1,d7  ; zwiększenie licznika linii
        dbf d5,PowielajPetla     ; pętla powielania bajtu
        bra.s    U2_Check        ; skok do sprawdzania
                                   czy został osiągnięty
                                   koniec linii
PadWord     dc.w  0 ViewModes     dc.w 0   ; PadWord
                  ; ViewModes to dane o obrazku pobrane
                  ze struktury CAMG
Height      dc.w  0              ; wysokość obrazka
Width       dc.w  0              ; szerokość obrazka
BytesPerRow       dc.w 0         ; ilość bajtów w linii
Depth       dc.b  0              ; ilość bitplane'ów
```

I to by było wszystko w dzisiejszym Kąciku Początkującego Kodera. Za miesiąc kolejne procedury.

Marcin "Duddie" Dudar

KURS JĘZYKA C

cz. 5

AMIGA

Dzisiaj pora na to, co stanowi fundament systemu operacyjnego Amigi - na struktury. Jak zwykle, zacznę od tego, że nie ma się czego bać. Struktura to nic innego jak grupa zmiennych mająca wspólną nazwę. Wyobraźmy sobie sytuację, w której chcemy otworzyć na ekranie okno. Do tego celu potrzebujemy szeregu parametrów, które to okno opisują: położenie lewego górnego rogu, szerokość i wysokość, typy gadżetów związanych z tym oknem, nazwę okna, minimalne i maksymalne rozmiary itp. Normalnie powołali byśmy do życia zmienne np. LeftX, LeftY, RightX, RightY, MinX, MinY itp. itd. Gdyby nasz program miał operować np. na pięciu oknach, wtedy parametry opisujące każde z nich musiałyby być w nowych zmiennych, np. LeftX1 itd. Program by wyglądał mniej więcej tak:

```
main()
{
        int  LeftX1, LeftY1, LeftX2, LeftY2, LeftX3, LeftY3,
             LeftX4 , LeftY4, LeftX5, LeftY5;
        int  RightX1, RightY1, RightX2, RightY2, RightX3,
             RightY3, RightX4, RightY4, RightX5, RightY5;
        .

/* I tak dalej, tysiace deklaracji, w ktorych bysmy sie
zgubili w kilka sekund po zadeklarowaniu. */
        .

 /* Tu dopiero pare linijek naszego programu */

        .
        .
};
```

Znacznie wygodniej jest użyć do tego typu problemów struktur. Przy operowaniu strukturami wyróżniamy trzy podstawowe fazy: definicja struktury, deklaracja struktury i operacje na strukturach. Zacznijmy od definicji (i od przykładu):

```
struct Okienko
{
        int LeftX, LeftY;
        int RightX, RightY;
        char *Name;
        int MinX, MinY;
        int MaxX, MaxY;
};
```

Słówko struct oznacza przystąpienie do definiowania (lub, jak to później zobaczymy, do deklarowania) struktury. Po struct następuje nazwa struktury. My zadeklarowaliśmy strukturę o nazwie Okienko. Co to dla nas oznacza? Oznacza to, że pojawił się nowy typ zmiennej. Składa się ona ze zmiennych wyszczególnionych w definicji pomiędzy nawiasami klamrowymi. Poszczególne elementy struktury - składowe struktury - są deklarowane tak samo, jak dotychczas zwykłe

zmienne. Zadeklarujemy teraz jakąś zmienną typu okienko:

```
struct Okienko Okno;
```

Mamy teraz jedną zmienną Okno, która zawiera kilka parametrów opisujących nasze wyimaginowane okno (UWAGA! System operacyjny Amigi używa do opisu okien struktur Window i NewWindow, ale nasz przykład nie ma z tym nic wspólnego!). Tak więc zamiast szeregu zmiennych mamy jedną.

Przejdźmy do przykładu. Do opisu okna użyjemy tylko czterech parametrów: LeftX, LeftY, RightX, RightY, które określałyby położenie lewego górnego i prawego dolnego rogu okna. Oto program:

```
/* Definicja struktury typu Okienko */
struct Okienko
{
        int LeftX, LeftY;
        int RightX, RightY;
};
/* Deklarujemy zmienna zewnetrzna typu Okienko */
struct Okienko Okno;

/* Wlasciwy program */
main()
{
        /* Nadajemy wartosci poszczegolnym skladowym */
        Okno.LeftX = 5;
        Okno.LeftY = 10;
        Okno.RightX = 50;
        Okno.RightY = 100;
 /* Drukujemy wartosci poszczegolnych skladowych */
  printf( "\nOto skladowe naszej struktury:\n" );
 printf( "\nLeftX=%d\nLeftY=%d\nRightX=%d\nRightY=%d\n",
     Okno.LeftX, Okno.LeftY, Okno.RightX, Okno.RightY);
};
```

Program składa się z 4 faz. Pierwsza to definicja struktury Okienko. Warto zaznaczyć, że definicja struktury nie przydziela pamięci, jest jedynie informacją dla kompilatora, jak ma traktować zmienne będące strukturami. Druga, to deklaracja zmiennej typu Okienko, trzecia to nadanie wartości poszczególnym składowym, a czwarta to po prostu wydrukowanie zawartości poszczególnych składowych struktury. Dotychczas jedynie definiowaliśmy strukturę i deklarowaliśmy zmienną będącą strukturą. Jak można zauważyć w programie, dostęp do odpowiedniej składowej następuje poprzez użycie operatora . (kropka). Odbywa się to według schematu:

```
        nazwa_struktury.nazwa_składowej
```

Tak więc linia

```
        Okno.LeftX = 5;
```

mówi: składowej LeftX należącej do zmiennej Okno (tu kompilator sprawdza: zmienna Okno jest deklarowana jako struktura Okienko, a struktura Okienko zawiera składowa o nazwie LeftX, więc wszystko OK) nadaj wartość 5. To wszystko. Czwarta faza to po prostu wydruk struktury na ekran.

Nadawanie wartości poszczególnym składowym może się odbywać w momencie deklaracji zmiennej (oczywiście tylko wtedy, gdy zmienna jest zewnętrzna!):

```
/* Definicja struktury typu Okienko */
struct Okienko
{
        int LeftX, LeftY;
        int RightX, RightY;
};
/* Deklaracja struktury i nadanie wartosci poszczegolnym skladowym */
struct Okienko Okno =
{
        5, 10, 50, 100
};

/* Wlasciwy program */
main()
{
/* Drukujemy wartosci poszczegolnych skladowych */
 printf( "\nOto skladowe naszej struktury:\n" );
printf("\nLeftX=%d\nLeftY=%d\nRightX=%d\nRightY=%d\n",
     Okno.LeftX, Okno.LeftY, Okno.RightX, Okno.RightY);
};
```

Jakie korzyści płyną z zastosowania struktur? Oprócz zwiększenia przejrzystości programu dzięki nim można przekazywać szereg parametrów pomiędzy funkcjami. Jak wiemy, funkcja zwraca tylko jedną zmienną. Jednakże struktura jest przecież JEDNĄ zmienną! W praktyce rzadko stosuje się zwracanie przez funkcje struktur, częściej zwracane są wskaźniki do struktur.

W poprzedniej części kursu była przedstawiona funkcja swap(). Argumentami funkcji swap() były dwa wskaźniki, sama zaś funkcja po prostu zamieniała wartościami zmienne wskazywane przez swoje argumenty. W tym sensie zwracała ona nie jedną, lecz dwie zmienne. I znowu, chęć zmiany większej ilości zmiennych powodowałaby konieczność tworzenia funkcji z dużą liczbą wskaźników jako argumentów, czego łatwo można uniknąć przekazując funkcji jedynie wskaźnik do struktury. Zanalizujmy to na przykładzie:

```
/* Definicja struktury typu Okienko */
struct Okienko
{
        int LeftX, LeftY;
        int RightX, RightY;
};
/* Deklaracja zmiennych */
struct Okienko Okno;
/* Funkcja nadajaca wartosci skladowym struktury Okienko */
void Wartosc Pierwsza ( struct Okienko *dana )
{
        dana->LeftX = 5;
        dana->LeftY = 10;
        dana->RightX = 50;
        dana->RightY = 100;
};
/* Funkcja nadajaca inne wartosci skladowym struktury Okienko */
void Wartosc Druga ( struct Okienko *dana )
{
        dana->LeftX = 60;
        dana->LeftY = 70;
        dana->RightX = 245;
        dana->RightY = 310;
};

/* Wlasciwy program */
main()
{
/* Nowe wartosci */
        Wartosc Pierwsza( &Okno );
 /* Drukujemy wartosci poszczegolnych skladowych */
 printf( "\nOto skladowe naszej struktury:\n" );
printf("\nLeftX=%d\nLeftY=%d\nRightX=%d\nRightY= %d\n",
 Okno.LeftX, Okno.LeftY, Okno.RightX, Okno.RightY );
 /* Nowe wartosci */
        WartoscDruga( &Okno );
```

```
/* Drukujemy wartosci poszczegolnych skladowych */
printf( "\nOto skladowe naszej struktury:\n" );
printf("\nLeftX=%d\nLeftY=%d\nRightX=%d\nRightY=%d\n",
  Okno.LeftX,  Okno.LeftY,  Okno.RightX,  Okno.RightY  );  };
```

W tym przypadku do nadawania wartości strukturze Okno używamy dwóch różnych funkcji, nadających tej strukturze różne wartości. Funkcje te (WartośćPierwsza() i WartośćDruga()) mają jako argumenty wskaźniki do struktur, tak więc wywołujemy je z argumentem poprzedzonym operatorem &:

```
WartośćPierwsza( &Okno );
WartośćDruga(  &Okno  );
```

Zwróćmy uwagę na zawartość funkcji WartośćPierwsza(). Pojawił się tam nowy operator: -> (minus, większe). Pamiętajmy, gdyż jest to bardzo istotne i należy na to zwracać OGROMNĄ uwagę:

Operator . (kropka) oznacza: "weź składową należącą do podanej struktury".

Operator -> (minus, większe) oznacza: "weź składową należącą do struktury wskazywanej przez wskaźnik".

Tak więc ilekroć chcemy się odwołać do składowej struktury należy zwrócić uwagę, jaki mamy dostęp do tej struktury. Jeżeli bezpośredni, np. struktura jest deklarowana jako zewnętrzna, wtedy używamy operatora . (kropka). Jeżeli dostęp do struktury mamy poprzez wskaźnik, używamy operatora -> (minus, większe).

Struktury są traktowane jak pozostałe zmienne proste (int, char itp.), tak więc możemy deklarować tablice struktur, tablice wskaźników do struktur itd. Nie wydaje mi się konieczne omawianie tego typu zabiegów, gdyż są to dokładnie takie same operacje jak na zwykłych tablicach, ograniczę się więc tylko do przykładów (pozwolą one oswoić się trochę z może dziwną na pierwszy rzut oka symboliką zapisu):

```
int  test[10];        /* tablica zmiennych typu int  */
test[1]  =  5;        /*  zmienna  test[1]  przyjmuje
                          wartość 5  */
int *test[10];        /* tablica wskaźnikow do
                          zmiennych typu int */
*test[1]  =  5;       /*  zmienna  wskazywana  przez
                          wskaźnik test[1] przyjmuje wartość 5*/
struct Okienko test[10];      /* tablica struktur typu
                          Okienko  */
test[1].LeftX  =  5;          /* składowa LeftX
                          struktury test[1] przyjmuje wartość 5*/
struct Okienko  *test[10];    /* tablica wskaźnikow
                          do struktur typu Okienko*/
    test[1]->LeftX = 5;       /* składowa LeftX
                          struktury wskazywanej przez test[1]
                          przyjmuje wartość 5   */
```

Sądzę, że na dziś starczy. Oswojenie się z informacjami z tej części kursu wymaga trochę czasu i praktyki. Jeśli nawet coś pozostaje niezrozumiałe po kilkakrotnym przeczytaniu, stanie się jasne po kilku próbach przy komputerze. Na tym kończę tą część cyklu. Następna część będzie stanowiła uzupełnienie informacji o podstawowych instrukcjach języka C.

Jarosław Chrostowski

PUBLIC DOMAIN PACK

AMIGA

Opis zestawu nr 13

Witamy w nowym PUBLIC DOMAIN PACKU. Co tym razem ujrzysz szczęśliwy jego użytkowniku? Ooo... bardzo dużo ciekawych programów!

INTERFERON PRO! nagrany na BOOTBLOCK'u jest po prostu rewelacyjny. Czego oni nie wpakowali do biednago malutkiego BOOT'a. Nawet kopiowanie dyskietek. No, ale przejdźmy do rzeczy. Po wyjściu z tego programiku ujrzycie spis zawartości dysku - wyboru programów dokonujemy przy pomoce „efów", tj. klawiszy funkcyjnych. Rozpoczynamy przeglądanie programów. START!

Naciskamy F1 i po chwili super demko! Węgierska grupa CERBEROS chyba niewiele ustąpiła PHENOMEN'ie. Podobny symulator lotu, świetna grafika i muzyka. To jest to! Może by tak i Polacy?

F2 -to już demo z zachodu. Dużo wektorów (nawet ze smugą punktów). SINUS SCROLL'e, perfekcyjne dopracowe!

F3 - oto bardzo ładna muzyczka skompnowana przez COSMOS'a z grupy ANARCHY. Zwyciężyła ona na COPY PARTY grupy DIGITAL. Sądzę, że głosujący mieli rację - jest naprawdę piękna.

F4, F5, F6 - po kolei: pierwsze, drugie i trzecie miejsce z konkursu na najlepszy obrazek (te samo COPY PARTY). Wygrał tam KREST - grafik również grupy ANARCHY. Swoją drogą zadziwia nas ta grupa swoją ilością tak dobrych artystów (grafików i muzyków).



F7. SUPER DUPER 2.01. Dosyć demosów, teraz poważne programy, a szczególnie ten - chyba najlepszy PUBLIC DOMAIN jakiego widziałem. Mimo nieciekawej nazwy jest - bez wątpienia - świetnym programem kopiującym. Podstawową jego zaletą jest praca w MULTI TASKINGU. Możemy na przykład uruchomić PRO TRACKER'a, a SUPER DUPER będzie kopiował sobie w międzyczasie dyski. Pewnie myślicie, że będzie to robił bardzo wolno, otóż nie, SUPER DUPER kopiuje dyskietki bez względu na to jaki program uruchomimy razem z nim o mniej więcej dziesięć sekund szybciej niż legendarny X-COPY. Często zdarza się, że nasze dyski „padną", ale czasami uda się komputerewi prawidłowo je odczytać.

AMIGA

X-COPY bez względu na to czy podczas odczytu wystąpił błąd czy też nie przystępuje do odczytu następnego track'u. SUPER DUPER nie ma takiego błędu. W okienku mamy możliwść wpisania ile razy program ma odczytywać zepsuty track. Po ustawieniu na maksimum (99) zepsuta ścieżka jest odczytywana z prędkością około 4 odczytów na sekundę i oczywiście czynność jest przerywana jeżeli odczyt będzie prawidłowy. Podobnie przy zapisie. Tym oto sposobem możemy odzyskać około 80% „padniętych" dysków, przekopiowując z nich programy na inne - bezpieczne. Nie są to oczywiście wszystkie zalety SUPER DUPER'a.

Teraz krótki opis najważniejszych opcji

SOURCE- wybór źródła z której stacji będziemy czytać.

DESTINATION- wybór napędu na który mamy nagrywać. Ciekawostką jest fakt, że nagrywanie na kilka napędów jednocześnie nie spowalnia zapisu.

KILL SYS - znany chyba z X-COPY (wyczyszczenie pamięci). Interesujące jest to, że po zrobieniu killsys pojawia się RESTORE powodujące przywrócenie poprzedniego stanu (działa jeśeli nie zużyliśmy na kopiowanie potrzebnej do tego pamięci RAM).

ELEPLEASED - wyświetlanie nazw ostatnio kopiowanych dziesięciu dysków. Niby banalne, a często przydatne.

SC i EC - wybór, od której do której ścieżki mają być kopiowane dyski.

RTRY:ERR - liczba błędów, które występowały przy odczycie: liczba błędów z jaką SUPER DUPER odczytał dysk.

LABEL - nazwa jaką chcemy dać formatowanemu dyskowi.

RETRY - ile razy mamy odczytywać zepsute ścieżki.

BUFFER - ustawienie RAM'u jako bufora, w którym mają być składowane informacje.

HDBUF - to samo z tym, że informacje będą składowane na twardym dysku.

VDBUF - w tym przypadku sami możemy ustawić sobie urządzenie jako bufor.

DATE - nadawanie każdemu kopiowanemu dyskowi aktualnej daty. Przydatne jeżeli mamy zegar.

INCNAME - Opcja pomocna przy seryjnym formatowaniu dysków. Polega na tym, że jeżeli w nazwie dysku (label) umieścimy 00 to przy następnym formatowaniu komputer zmieni to na 01 itd.

TALK - po włączeniu tej opcji komunikaty będą wypowiadane przez komputer głosem ludzkim (na może amigowym).

AUTO - włączenie powoduje, że po każdorazowym włożeniu dysku do źródła i przeznaczenia komputer automatycznie skopiuje dyskietkę (dzięki temu możemy zająć się w między czasie innym programem, spoglądając tylko od czasu do czasu na diodę w stacji lub słuchając sygnałów dźwiękowych).

STOP - przerwanie wykonywanej aktualnie operacji (jak w X-COPY).

GO- rozpoczęcie nagrywania (w przypadku wybrania na źródło i cel różnych stacji - rozpoczęcie kopiowania).

READ - odczyt dyskietki do bufora (trzeba go najpierw ustawić).

CHECK - sprawdzenie dysku.

FORMAT - formatowanie dyskietki.

AREXX - włączenie współpracy z systemem AREXX.

To by było na tyle. Dodam jeszcze, że SUPER DUPER w przypadku kopiowania przez bufor włącza automatycznie kompresję danych gromadzonych w pamięci, nie spowalniając odczytu (przynajmniej niezauważalnie).

Na dysku zamieszczona jest również oryginalna instrukcja napisana przez autora programu (zajmująca ponad 20 kB).

F8 - **SANITY COPY**. Program zakodowany przez CHAOS'a znanego chociażby z dema ELYSIUM. Kopier nie jest tak rozbudowany jak - znany wszystkim - X-COPY, ale za to jest bardzo szybki (podobnie jak SUPER DUPER), ma możliwość kodowania dysków i analizy bootblocku. Bardzo wygodne jest też wybieranie ścieżek do kopiowania i kopiowanie tylko zepsutych (ponowna próba). Rewelacyjne są również odgłosy (naprawdę stawiają na nogi) - przekonajcie się sami!

F9 - **NOISE PACKER 3.00**. Program przeznaczony właściwie tylko dla koderów. Nowy standard zapisu muzyki stworzony przez TWINS'a z PHENOMEN'y. Program konwertuje normalne moduły z NOISE, STAR, PROTRACKER'a itp. na nowy NP. Cały fenomen tego zabiegu polega na tym, że play routine do muzyczek „znoisepackowanych" zabiera komputerowi w czasie odtwarzania muzyki trzykrotnie mniej czasu niż narmalny. Dlatego też w każdym nowym dobrym demie począwszy od ENIGMA/PHENOMENA muzyka jest w tym formacie. Do programu dołączone są trzy programy źródłowe (play routine) w podkatalogu **SOURCES.**

F10 - **HAM SHARP** - program konwertuje obrazki z formatu GIF na IBM'ie na IFF na Amidze. Po wgraniu programu podajemy jedynie nazwę pliku w formacie GIF i docelowego IFF w HAM-ie. Dzięki temu programowi możemy ściągnąć zdjącia, z których tak dumni są posiadacze IBM'a.



MOSTRA (M, parametry) - program wyświetlający obrazki w IFF. Jego autorem jest SEBASTIANO VINGA - autor SUPERDUPER'a. Jeżeli chcecie zobaczyć o jakie parametry chodzi to wpiszcie po prostu M (i return oczywiście). Mogę was jednak zapewnić, że z pewnością jest to najlepszy program tego typu.

I to by było na tyle... Za miesiąc możecie się spodziewać czegoś jeszcze lepszego, a pack powinien wyglądać nieco lepiej graficznie. Niezapomnijcie więc nabywać gc regularnie.

NINJA/ATD

Najbardziej rozpowszechnioną w Polsce kartą do AMIGI jest w tej chwili z pewnością ACTION REPLAY v2.0 - Jednak ostatnio pojawiło się jeszcze jedno tego typu urządzenie o nazwie X-POWER PROFESSIONAL. Obydwa przeznaczone są do AMIGI 500 i podłączane z jej lewej strony do szyny systemowej.

AMIGA

ACTION REPLAY v2.0 czy X-POWER PROFESSIONAL?

Na początek kilka słów na temat wyglądu zewnętrznego. Z pewnością ładniej prezentuje się ACTION REPLAY v2.0 (zwany dalej AR) - czarno-szare pudełko z ładnie wmontowanym potencjometrem i przełącznikiem do spowalniacza oraz estetycznie wpasowanym przyciskiem freeze. Na karcie znajdują się również dwie diody informujące o działaniu karty oraz samego spowalniacza. Wszystko to robiłoby świetne wrażenie gdyby nie za bardzo wystająca z obudowy płytka.

Wadą, bardzo uciążliwą, jest brak przelotu szyny systemowej, przez co można mieć spore trudności z podłączeniem innego urządzenia nie wyposażonego w przelot szyny. Natomiast X-POWER PROFESSIONAL (zwany dalej XP) mimo dopasowanej kolorystycznie obudowy po prostu źle wygląda. Przycisk freeze wystaje z obudowy i jest krzywo umieszczony, potencjometr spowalniacza jest praktycznie niedostępny dla użytkownika ze względu na bardzo małe wymiary. Również fatalnie umieszczony jest wyłącznik spowalniacza: w dziurze wyciętej z tyłu obudowy karty. W sumie wygląda to bardzo nieestetycznie. W XP wbudowane są trzy diody LED informujące o trybie pracy karty.

Warto wspomnieć, że XP ma przelot szyny systemu, co jest jego dużą zaletą. Jednakże nie wszystkie urządzenia dają się w nie włożyć ze względu na bardzo wąską szczelinę.

Wadą XP jest brak wydruku instrukcji. Jest ona umieszczona na dysku dołączonym do karty. Wadę tę przynajmniej częściowo rekompensuje możliwość prawie natychmiastowego wyświetlenia jej bez potrzeby wychodzenia z działającego w danej chwili programu.

Teraz o działaniu kart. Po włączeniu AR zgłasza się całkiem ładnym napisem, natomiast XP swoją obecność sygnalizuje tylko zapaleniem zielonej diody. Na pewno plansza tytułowa AR jest z początku całkiem przyjemna, jednak po dłuższym czasie zaczyna denerwować. Działanie spowalniacza pracy procesora jest w obu kartach takie samo, jednak wygoda obsługi w AR jest znacznie większa. Główną funkcją tych kart jest możliwość zatrzymywania każdego programu w dowolnym momencie. Po naciśnięciu przycisku umieszczonego w obudowie kart program zostaje zatrzymany, a karta uruchamia swój własny program. W AR ukazuje się pusty ekran i kursor. Bez instrukcji, albo przynajmniej help'a się nie obejdzie.

Naciskamy więc klawisz HELP i ukazuje się spis wszystkich dostępnych poleceń wraz z bardzo krótkim opisem ich działania i składnią, za co należy się duży plus. W XP natomiast pojawia się najpierw obrazek jaki pozostał na ekranie po zatrzymaniu programu, a na nim okienko zawierające z lewej strony spis dostępnych opcji, w prawym dolnym rogu zegar, a w prawej górnej części okienka znajduje się miejsce na rozwijanie wszystkich opcji z głównego MENU. Także XP posiada swój HELP, dostępny po wywołaniu odpowiedniej opcji (funkcje wybierane za pomocą klawiatury).

W dalszym opisie będę się kierował układem opcji XP.

■ Pierwsza pozycja w MENU XP to „GRAFIX".

Po wybraniu jej na aktualnym obrazku pokazuje się okienko zawierające następujące informacje:

- położenie kolejnych BITPLANE'ów,
- pamięci koloru oraz okienka na ekranie,
- wysokość obrazka,
- ilość BITPLANE'ów składająca się na dany obrazek,
- opcja przejścia do edycji kolorów.

W edycji kolorów należy wybrać, który z nich ma być zmieniony i używając suwaków RGB odpowiednio go ustawić.

Ponad to istnieje możliwość przełączania trybów INTERLACE, HAM, HI-RES, LO-RES i DUAL-PLAYFIELD.

Oczywiście odpowiednio ustawiony obrazek można nagrać na dysk lub wydrukować.

Teraz przechodzimy do AR. Po wpisaniu rozkazu „P" ukazuje się ekran, taki jaki był w momencie zatrzymania programu. Można teraz przeszukiwać inne obrazki, zmieniać kolory, tryby wyświetlania obrazu itp. Wszystko to jednak odbywa się za pomocą klawiatury, a informacje o adresach BITPALNE'ów itp. nie są wyraźnie przedstawione. Bez instrukcji się nie obejdzie, przynajmniej na początku.

■ Przechodzimy do następnej funkcji: SPRITES.

XP umożliwia edycję sprite'ów oraz dowolne ich modyfikowanie: jak położenie w pamięci, wysokość, rodzaj, a także edycję samego sprite'a. AR ma podobne możliwości jednak jest znacznie mniej wygodny w obsłudze.

AMIGA

■ **Kolejną funkcją w XP jest wyszukiwanie muzyki i sampli.**

W AR te dwie opcje są zupełnie oddzielone.

W XP przy wyszukiwaniu sampli można zmieniać ich początek, koniec oraz prędkość ich odtwarzania. Przy wyszukiwaniu modułów możliwe jest ich odtworzenie, wraz z pokazaniem aktualnie odtwarzanego pattern'u. Można też odtwarzać tylko jeden wybrany kanał. Wyszukiwanie modułów w pamięci jest dość szybkie. AR posiada bardzo dobry system wyszukiwania sampli.

Po wydaniu rozkazu SCAN można rownież zmieniać granice sampla, jak i szybkość jego odtwarzania.

Rozkaz TRACKER służy do rozpoczęcia (czasem bardzo długo trwającego) poszukiwania modułu. Gdy moduł zostanie znaleziony można go odtworzyć, zobaczyć spis instrumentów, wszystkie ważniejsze dane, a także zmienić jego format na moduł 16-instrumentowy. Oczywiście w przypadku obu kart można nagrać na dysk zarówno sample, jak i moduły.

■ **Kolejną opcją w menu głównym XP jest monitor kodu maszynowego.**

Nie będę tu szczegółowo omawiał jego możliwości, gdyż zajęłoby to zbyt dużo miejsca. XP ma jednak nieco mniejsze możliwości niż AR.

Obydwie karty posiadają takie komendy, jak:

- deassemblacja,
- assemblacja,
- deassembler copper'a,
- poszukiwanie danych (AR również znaków),
- wypełnianie pamięci,
- przerzucanie bloków pamięci,
- zgrywanie danych,
- pokazywanie pamięci w formie znaków ASCII,
- pokazywanie najważniejszych danych systemowych,
- czytanie ścieżek z dysku.

■ **Kolejnymi pozycjami w MENU karty XP są IMPORT oraz EXPORT.**

Służą one do zgrywania na dysk pamięci w formie spakowanej (AR,XP) lub też nie spakowanej (XP).

■ **Następną opcją jest DOS.**

XP oraz AR również tutaj mają bardzo podobne możliwości. Oczywiście w AR rozkazy muszą być zawsze wpisywane z klawiatury, natomiast w XP są wybierane myszą lub klawiszami strzałek.

W obu kartach są:

- kasowanie pliku,
- kopiowanie całego dysku,
- tworzenie katalogu,
- formatownie dysku,
- zmiana bootblock'u.

XP umożliwia: zmianę nazwy pliku, kopiowanie plików, AR umożliwia ponad to zaś: kopiowanie dysku z kodowaniem, kodowanie bootblock'u, sprawdzanie błędów na dysku.

■ **Kolejna funkcja XP to menu z różnymi przydatnymi „pomocami".**

- Pierwszy to AUTOFIRE.
  XP posiada możliwość ustawienia 255 różnych częstotliwości strzelania. W AR AUTOFIRE uruchamia się z menu set-up (F3), gdzie również można ustalić częstotliwość strzelania oraz port, w którym instalujemy AUTOFIRE.
- Druga opcja to dorabianie trainerów do gier, która w obu kartach działa podobnie, jednak w XP ma nieco prostszą obsługę.
- Trzecią opcją w XP jest wczytanie instrukcji obslugi z dysku (nie dotyczy AR).
- Kolejna opcja to slideshow, czyli pokaz obrazków. Umożliwia ona zrobienie własnego pokazu obrazków. AR takiej możliwości nie ma.
- Następna to sprawdzanie bootblock'u dysku znajdującego się w stacji. Obydwie karty wykryły dwa wirusy jakie udało mi się wygrzebać z dysków: LAMER EXTERMINATOR i BYTE BANDIT. Cartridge usuwają również wirusy z pamięci. AR przy próbie boot'owania dysku z podejrzanym bootblock'iem zgłasza odpowiedni komunikat przerywając booting i przechodząc do menu, w którym jest możliwość zmiany wektorów reset'u, instalowania nowego bootblock'u, wyświetlenia aktualnego bootblock'u i przejścia do monitora.
- Kolejna opcja w XP to sprawdzenie działania joystick'a.
  Nie jest to chyba najpotrzebniejsze narzędzie. AR go nie ma.
- Dalej w XP jest czasem przydatna opcja zmiany nazwy dysku.
  AR również jej nie ma.
- Kolejną opcją jest wyłączenie lub włączenie filtrów (AR nie ma).
- Oraz, na koniec, możliwość wybrania jednego z czterech standardów klawiatur: szwedzki, francuski, ASCII oraz DIN.
  AR ma możliwość wyboru pomiędzy klawiaturą niemiecką, a amerykańską (angielską).

XP jako dodatek ma również wbudowany program X-COPY v3.2. Można go w każdej chwili wywołać, jednak powoduje to utratę wszystkich danych z pamięci.

Oprócz tego XP ma jeszcze chyba kilka drobnych błędów, gdyż dwa razy w ciągu jednego dnia odmówił odwieszenia komputera nawet po resecie i w końcu musiałem wyłączyć komputer z sieci. W przypadku AR w ciągu już dość długiego czasu eksploatacji nic takiego jeszcze się nie zdażyło.

Podsumowując muszę stwierdzić, że nie zamieniłbym AR na XP. Mimo dużo łatwiejszej obsługi, XP jest gorzej zaprojektowanym cartridgem. Jest on raczej przeznaczony dla początkujących użytkowników AMIGI. Ktoś mógłby powiedzieć, że zaawansowany programista nie musi używać takich pomocy, jak AR czy XP - nie zgadzam się z tym poglądem. W jakim celu człowiek ma się niepotrzebnie męczyć nad czymś godzinę, jeśli może wykonać tą samą robotę w ciągu kilku minut, a nawet sekund?

Jarri

W dzisiejszym odcinku zmagań z arp.library zajmiemy się kolejnymi procedurami tej użytecznej biblioteki.

AMIGA

ARP LIBRARY cz.4

CheckSumPrg - **obliczenie sumy kontrolnej dla programu rezydentnego**

SumaKontrolna =**CheckSum (Node)**

D0 **D1**

Suma kontrolna wszystkich programów w liście programów rezydentnych jest obliczana gdy są one dołączane do listy. W niektórych przypadkach może okazać się konieczne przeliczenie na nowo sumy kontrolnej. Funkcja ta nie zmienia wartości node i dlatego jest jedyną funkcją, której należy używać. Oto poprawne metody użycia tej funkcji:

-dla języka C:

```
Node = ObtainResidentPrg("Nazwa");
SumaKontrolna = CheckSumPrg(Node);
ReleaseResidentPrg(Node-rpn_Segment);
```

-dla asemblera:

```
        move.l  ArpBase,a6
        lea     Name,a0
        jsr     ObtainResidentPrg(a6)
        move.l  d0,d1
        move.l  d0,Node
        jsr     CheckSumPrg(a6)
        move.l  Node,a0
        move.l  rpn_Segment(a0),d1
        jsr     ReleaseResidentPrg(a6)
Node    dc.l 0
Name    dc.b    'Nazwa',0
```

Wejście:

Node - wskaźnik node dla programu rezydentnego.

Wyjście:

SumaKontrolna - nowa suma kontrolna dla programu rezydentnego.

GetEnv - **pobranie wartości ze środowiska zmiennych**

Wskaźnik = **GetEnv("Ciąg", Bufor, Długość)**

D0 **A0** **A1** **D0**

Ta funkcja wprowadza mechanizm środowiska zmiennych kompatybilny z MANX'em. Jest bardziej efektywna niż funkcja getenv() z manx'a.

Wejście:

Ciąg - wskaźnik nazwy środowiska zmiennych.

Bufor - obszar przydzielony przez użytkownika, wykorzystywany do wprowadzenia wartości przypisaną zmiennej. Może być NULL (0).

Długość - rozmiar bufora (w bajtach) gdzie będzie wprowadzona zienna.

Wyjście:

Wskaźnik - jeżeli znaleziono zmienną to jest to wskaźnik ciągu wartości dla tej zmiennej. Jeżeli bufor jest podany to będzie do niego skopiowany ciąg wartości o długości dopuszczalnej przez długość bufora użytkownika.

StampToStr - **dokonuje konwersji struktury DateStamp jako części struktury DateTime do ciągu ASCII**

Błąd = **StampToStr(DateTime)**

Procedura ta dokonuje konwersji AmigaDOS-DateStamp struktury do ciągu znaków ASCII w sposób jaki zarządamy w strukturze DateTime poprzez podanie odpowiednich parametrów.

Wejście:

DateTime - wskaźnik zainicjowanej struktury DateTime.

Struktura DateTime wygląda następująco.

```
STRUCTURE   DateTime,0
  STRUCT  DateStamp,ds_SIZEOF  ;kopia struktury Date
                                Stamp, która ma być
                                poddana konwersji
  UBYTE   dat_Format           ; parametry kontrolne
                                wyprowadzanej daty
  UBYTE   dat_Flags            ; znaczniki dla wyprowadzanego
                                dnia
  CPTR    dat_StrDay           ; bufor na nazwę dnia tygodnia
                                (Monday, etc...)
  CPTR    dat_StrDate          ; bufor na datę
  CPTR    dat_StrTime          ; bufor na czas
```

Każdy bufor musi być o wielkości 10 znaków.

Znaczniki dat_Flags:

```
DT_SUBST  =1      ; wpisanie zamiast nazwy dni
                    tygodnia nazw zastępczych
                    np. Today (Dzisiaj), Tommorow
                    (Jutro), etc.
DT_FUTURE=2       ; jeżeli data wypada w przyszłości
                    wpisanie Future (Przyszłość)
                    zamist nazwy dnia tygodnia.
```

Znaczniki dat_Format (sposób zapisu daty):

```
FORMAT_DOS=0      ; format AmigaDOS (dd-mmm-yy)
FORMAT_INT=1      ; format międzynarodowy
                    (yy-mmm-dd)
FORMAT_USA=2      ; format amerykański (mm-dd-yy)
FORMAT_CDN=3      ; format kanadyjski (dd-mm-yy)
```

Wyjście:

Błąd - jeżeli wartość jest różna od zera to oznacza, że struktura DateStamp była podana błędnie i nie może nastąpić konwersja.

AMIGA

StrToStamp - **konwersja ciągu znaków zawartch w strukturze DateTime do struktury DateStamp.**

Błąd =StrToStamp(DateTime)

Dokonuje konwersji ciągu znaków ASCII podanych w strukturze DateTime do struktury DateStamp zawartej w strukturze DateTime. Struktura DateTime powinna być zainicjowana następująco:

ustawiony znacznik dat_Format dla zadanej daty jak dla stuktury DateTime używanej przez StampToStr, znaczniki dat_Flags dla poprawnego odczytania dat_StrDate oraz oczywiście bufory dat_StrDate, dat_StrDay, dat_StrTime.

Wejście:

DateTime - struktura DateTime z zainicjowanymi buforami oraz znacznikami

Wyjście:

Błąd - zero oznacza, że struktura DateStamp zawiera odpowiednio przetworzone daty oraz czas, natomiast gdy procedura zwróci wartośc niezerową oznacza to, że konwersja nie może być dokonana.

Marcin "Duddle" Dudar

Komputerowe rozmaitości

Każdy z nas będąc choć raz za granicą doznał jakiegoś szoku, porównując sytuację w kraju ojczystym i obcym. Chciałbym podzielić się wrażeniami z mojej ostatniej podróży do Francji. Sam kraj uważany jest przez jego mieszkańców za centrum przyszłej zjednoczonej Europy i daje się to odczuć patrząc na poziom życia i oferowanych tam usług. Nie będę się tu rozwodził nad ilością dostępnych gatunków sera w sklepach, czy rodzajach mleka, lecz nad bogactwem rynku komputerowego (a w szczególności komputerów domowych). Co możemy dostrzec wchodząc do pierwszego z brzegu salonu komputerowego (tak, tak salonu, a nie sklepu). Otóż nie znajdziemy w nim tak popularnych kiedyś Spectrum'ów, ani małych Atari nie mówiąc już o Amstradach (można je nabyć chyba tylko drogą wysyłkową). A co możemy pooglądać (lub nabyć jeśli mamy zasobną kieszeń)? Przede wszystkim wszelkiego rodzaju klony IBM'a (od dużych stacjonarnych tower'ów do przenośnych cudeniek), przy czym prym wiedzie tu firma Commodore. Oprócz tego możemy kupić każdy rodzaj Amigi (500, 500 Plus, 3000,CDTV). Tak jak wszędzie na zachodzie mamy możliwość dokonania dokładnego testu sprzętu, który chcemy kupić. W ten sposób zwykły, szary człowiek ma okazję pobawienia się Amigą 3000, lub innym komputerem. Lepsza zabawa zaczyna się w momencie gdy poprosimy sprzedawcę o zaprezentowanie paru programów (najczęściej gier). Jeśli ma się trochę oleju w głowie to można spowodować dość komiczny zamęt. Zilustruję to przykładem.

Sprzedawca: Czym mogę służyć?

RT: Czy może mi pan zademonstrować kilka programów?

S: Ależ oczywiście. Jakie gry chciałbyś zobaczyć? (Jeśli wyglądasz na małolata to z miejsca przechodzą z tobą na „ty" i proponują ci gry)

RT: Jeśli chodzi o gry to może coś z nowości?

Sprzedawca na moment znika by pojawić się z kilkoma kolorowymi pudłami. Rozpoczyna się pokaz pierwszego programu. W tym momencie zauważam, że gra się nie „odpali", gdyż nie współpracuje z kickstart'em 2.0 (do demonstracji używna była Amiga 500 Plus).

RT: Chyba nie zobaczymy tej gry?

S: To niemożliwe! Na tym nowym modelu działają wszystkie gry.

Tak jak przewidziałem „giera" się zablokowała. Sprzedawca spostrzegł już swoją pomylkę i próbuje z twarzą wybrnąć z sytuacji.

S: Dyskietka musi być uszkodzona i stąd te problemy. Niestety jest to jedyny egzemplarz.

Cwaniak! Zobaczymy co będzie dalej. Tym razem wybrał grę, na której jest naklejka świadcząca o kompatybilności z nową Amigą.

Po około 20 minutach odwracam się w stronę CDTV'ki i proszę o zademonstrowanie jej możliwości. Sprzedawca lekko drżącym głosem odpowiada, że nie jest w stanie tego zrobić, gdyż sprzęt ten ma u siebie dopiero od kilku dni. Wobec tego sam podchodzę do komputera i zaczynam się nim „bawić". Kilka podstawowych czynności podpatrzyłem na Commodore Expo w Warszawie i teraz mogę to zaprezentować ku coraz większemu zdumieniu stojącego obok sprzedawcy. Po chwili zabawy z dyskiem dodawanym do komputera zaczynam się nudzić i dokładniej przyglądać otoczeniu. Jest! W samym końcu sali zauważam eleganckiego IBM'a. Po krótkiej inspekcji okazuje się, że mam do dyspozycji kartę graficzną Super VGA. Wgrywam szybko „Scret of Monkey Island 2", by nasycić swą ciekawość i w tym momencie pada pytanie, którego spodziewałem się od początku.

S: Ty to chyba nie jesteś Francuzem?

RT: Nie. Jestem z Polski.

S: To tam macie w domach takie komputery?

RT: Jak widać mamy.

Chiałem jeszcze dodać, że my nie tylko gramy w gry, ale mógł by to potraktować jako obelgę. Ja natomiast nie miałem ochoty zepsuć sobie opinii w miejscu, które zamierzałem zamiar odwiedzać codziennie w czasie mojego miesięcznego pobytu. Po około 2 godzinach moich zmagań z najnowszymi grami stwierdziłem, że od dłuższego czasu sprzedawca przygląda mi się.

Na drugi dzień, gdy pojawiłem się w tym samym sklepie z miejsca wylądowała przy mnie sterta świeżych programów.

Na koniec wypada podsumować ten krotki felieton. Jeśli macie zamiar spędzić mile czas w sklepach komputerowych za granicą musicie przestrzegać pewnych podstawowych zasad.

Po pierwsze nie należy zdradzać pochodzenia przed pokazaniem swoich umiejętności (na zachodzie w dalszym ciągu popularny jest obraz komputerowej pustyni na wschód od Odry).

Po drugie należy zachowywać się uprzejmie, aczkolwiek stanowczo jeśli chodzi o rozmowę z wszelkiego rodzaju sprzedawcami.

Po trzecie należy upatrzyć sobie jeden sklep i eksploatować go do końca, gdyż wzbudza to zaufanie właściciela.

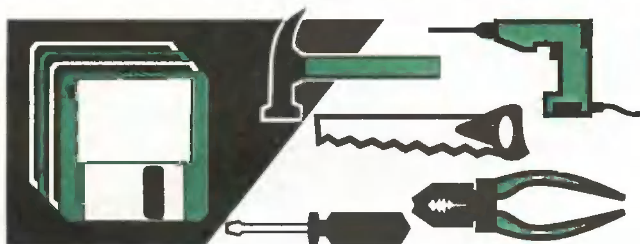
Robert "Mr.Raf" Turliński

Firma Prograsive Peripherals & Software zasłynęła w roku 1987 programem Disk Master v1.0 autorstwa Grega Cunninghama. Program ten, jak na owe czasy był wspaniałym „użytkiem" służącym zarówno do zabawy ze zbiorami, jak i ich kopiowania. Bił na głowę wszelkie programy tego typu (na przykład CLImate).
Płynęły lata, powstawały nowe wersje programu...

AMIGA

DISK MASTER V2.0

Najpierw dotarła do nas wersja 1.3, później - bardzo dobra - wersja 1.4 (opisywana w „64+4"), która nie była jednak wolna od błędów. Były także próby stworzenia własnych Disk Master'ów - jak to uczynił Gully, lecz do perfekcji oryginału brakowało wiele. Po czterech latach przerwy pan Greg Cunningham pokazał, że to On jest prawdziwym autorem Disk Master'a i tylko On wie czego potrzebuje program kopiujący roku 1991.



Nowy Disk Master, tym razem oznaczony jako wersja 2.0, to rewolucja w dziedzinie programów kopiujących. Wszelkie nowe „podróbki" typu: File Master czy Disrectory Opus wypadają bardzo słabo przy Disk Masterze 2.0.

Autor wprowadził w swoim programie znaczne zmiany, i tak na przykład nie mamy stałej wielkości dwóch okien oznaczonych S(ource) i D(estination), ale możemy mieć kilkanaście okien z różnymi katalogami, które chowamy pod siebie, wyciągamy na wierzch, przesuwamy po ekranie i wybieramy z nich zbiory, a tylko dwa z nich oznaczamy symbolami S i D. Także okno komend - C, jest ruchome i można zmieniać jego wielkość, a nawet je zamknąć.

Okien z komendami możemy mieć kilka - jeżeli ktoś naprawdę lubi. Ale najważniejsze jest to, że Disk Master jest całkowicie programowalnym programem, a program dla niego piszemy definiując plik konfiguracyjny. Możemy tworzyć własne komendy, wprowadzać własne menu, definiować wielkości i ilości okien, kolory a nawet wygląd linii z nazwami programów (np. czy chcemy mieć godzinę przed datą czy za datą i czy data ma być na przkład zapisana w formacie dzień-miesiąc-rok czy odwrotnie, a może ze spacjami zamiast kresek). Można nawet wyedytować linię z nazwą programu.

Jak już wspomniałem wszystkiego dokonujemy w pliku konfiguracyjnym, który nazywa się w Startup.DM i powinien znajdować się w katalogu S. Dzięki temu plikowi możemy dokonać na przykład spolszczenia programu Disk Master, ale komu to potrzebne? Możemy więc Disk Master'a rozbudowywać o wiele nowych funkcji, lub korzystać z jego własnej pokaźnej biblioteki.

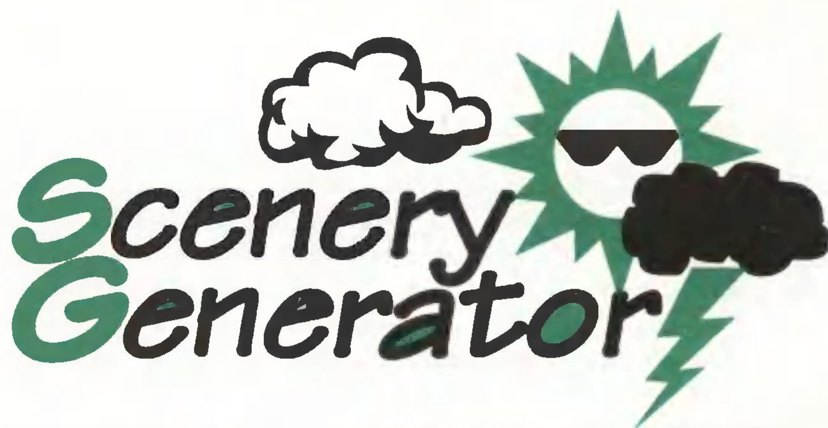
W tej wersji autor poprawił procedurę odczytywania tekstu - działa w 100 procentach poprawnie i jest dosyć wygodna w użyciu.

Poprawiona została także procedura wyświetlania obrazków. Niestety autor poprawiając stare błędy uczynił wiele nowych. I tak na przykład procedury formatowania nie działają dla nakładki na AmigaDOS symulującej IBM'a ani nawet dla rezydentnego ramdysku (RAD:), który jest w stu procentach kompatybilny ze zwykłym dyskiem. Także wczytywanie RAM'u spod Disk Mastera zakończy się zawieszeniem systemu - należy wyjść do CLI wczytać RAM: a dopiero potem rozpocząć pracę. Przyczyna leży nie wiadomo w czym, ale jest to dosyć uciążliwe.

Pomijając jednak te niedogodności należy stwierdzić, że jak na razie nie było lepszego programu kopiującego i chyba długo nie będzie (chyba że wersja 2.1 lub wyżej).

Marcin „Duddie" Dudar

AMIGA



Scenery Generator jest programem służącym do tworzenia krajobrazów. Jego największą zaletą jest fakt, że do uzyskania naprawdę pięknych efektów nie są potrzebne żadne umiejętności malarskie. Wystarczy tylko trochę popróbować... następnie poczekać kilka minut i już mamy gotowy, wspaniały krajobraz. Można tworzyć krajobrazy górskie, pustynne, morskie, zimowe i wiele, wiele innych. Możliwości są praktycznie nieograniczone. Na wykonanie takiego samego obrazka pod np. Deluxe Paint'em trzeba by poświęcić około 10 godzin (a do tego jeszcze trzeba umieć się nim posługiwać).

Scenery Generator praktycznie nie wymaga od nas nic, poza znajomością obsługi programu. Na początku ustalamy takie parametry, jak: wysokośc lądu, poziom wody, kierunek padania światła (doskonale znane każdemu „amigantowi" pull down menu).

A oto opis poszczególnych menu:

PROJECT:

About - o autorach.
Preview - pokazanie poglądowego planu naszego krajobrazu.
Med detail - rozpoczęcie tworzenia zdjęcia w niskiej rozdzielczości.
Hi detail - rozpoczęcie tworzenia zdjęcia w wysokiej rozdzielczości.
Workbench - przełączenie na workbench.
Load - załadowanie obrazka i ustawienia opcji.
Save - zapis obrazka na dysk, save posiada następujące podopcje:
settings - zapisane zostają same opcje.
settings+320*200 picture - opcje plus obrazek 320 na 200.
settings+352*240 picture - opcje plus obrazek 352 na 240.
settings+384*240 picture - opcje plus obrazek 384 na 240.
Obrazki zostaną zapisane w popularnym formacie IFF.
Print - wydrukowanie obrazka na drukarce.
Abort - przerwanie tworzenia obrazka.
Quit - wyjście z programu.

LAND:

Random seed - losowe wybranie ukształtowania terenu.
Enter seed - ukształtowanie terenu wybiera użytkownik.
Land height - ustalenie wysokości terenu.
Include brown - wprowadza kolor brązowy (będzie to przypominać drzewa).
Include gray - wprowadza kolor szary (piasek,itp.).
Include green - wprowadza kolor zielony (trawa,listowie).
Include snow - wprowadza śnieg (szczególnie efektownie wygląda śnieg na obrazku przedstawiającym bardzo wysokie góry).
Green level - ustawienie poziomu zieloności.
Snow level - ustawienie poziomu śniegu.

LIGHTING:

Ta opcja umożliwia nam ustawienie kierunku padania światła słonecznego.

Left front - słońce z przodu,nieco po lewej.
Front - słońce z przodu.
Right front - słońce z przodu,nieco po prawej.
Left - słońce po lewej.
Overhead - zenit.
Right - słońce po prawej.
Left back - słońce z tyłu,nieco po lewej.
Back - słońce z tyłu.
Right back - słońce z tyłu,nieco po prawej.

WATER:

Water - znacznik,który określa czy w ogóle chcemy mieć wodę.
Water level - ustawienie poziomu wody.
Add texture - czy chcemy tło? (niebo).
Beaches - określa sposób rysowania brzegów, jeżeli jest włączony, to komputer będzie rysował plaże.

OPTIONS:

Clouds - włączamy, jeśli chcemy mieć chmury (bardzo ładne)
Modify palette - zmiana kolorów obrazka, cały obrazek jest w 32 kolorach, z czego 4 to kolory nieba, 4 to kolory wody i 24 kolory lądu.
Restore palette - przywraca starą paletę.

To już wszystkie opcje Scenery Generator'a. Gorąco polecam ten program wszystkim, których interesuje grafika komputerowa. Życzę wielu owocnych godzin spędzonych na tworzeniu ciekawych rysunków.

Michał Gosztyła (Amber)

P.S. Zapraszamy do nadsyłania swoich prac graficznych do naszej redakcji. Najciekawsze będziemy starali się opublikować. Do dyskietki prosimy dołączyć opis zawierający imię, nazwisko i dokładny adres nadawcy oraz nazwę programu, którego użyliście do stworzenia „obrazka" (niekoniecznie musi to być Scenery Generator). Dyskietki będą przez redakcję zwracane.

Jeśli poszukujesz ciekawej literatury o Twoim komputerze to





kup ROCZNIK

64 PLUS 4 & AMIGA

Ładnie oprawiony tom zawiera numery od listopada 1990 r. do grudnia 1991 r.

Aby stać się jego posiadaczem wystarczy wpłacić 70 tys. zł
(w cenę wliczono koszt przesyłki)
na konto: Bank PKO SA Bydgoszcz,
konto nr: 5.09011-400522.7-136-11-111.0.
Na blankiecie wpłaty prosimy dopisać: "ROCZNIK"